SunplusIT

SPMC75X 编程指南

V1.7 – Nov. 14, 2005

6F,No.12, Lising Road • Science Park • Hsin-Chu • Taiwan 300 • R.O.C.

Tel: 886-3-563-2822 Fax: 886-3-668-7799 E-mail: mcu@Sunplus.com.cn

http://www.Sunplusmcu.com http://mcu.Sunplus.com

版权声明

凌阳创新科技股份有限公司保留对此文件修改之权利且不另行通知。凌阳创新科技股份有限公司所提供之信息相信为正确且可靠之信息，但并不保证本文件中绝无错误。请于向凌阳创新科技股份有限公司提出订单前，自行确定所使用之相关技术文件及规格为最新之版本。若因贵公司使用本公司之文件或产品，而涉及第三人之专利或著作权等智能财产权之应用及配合时，则应由贵公司负责取得同意及授权，本公司仅单纯贩售产品，上述关于同意及授权，非属本公司应为保证之责任。又未经凌阳创新科技股份有限公司之正式书面许可，本公司之所有产品不得使用于医疗器材，维持生命系统及飞航等相关设备。

目 录

第 1 章 概述 14

1.1 简介 14
1.2 芯片特性 14
1.3 应用领域 15

第 2 章 结构概述 16

2.1 简介 16
2.2 芯片结构 16
2.2.1 SPMC75F2413A 内部结构图 16
2.2.2 SPMC75F2313A 内部结构图 17
2.2.3 硬设资源表 17
2.3 芯片封装 18
2.3.1 SPMC75F2413A 18
2.3.2 SPMC75F2313A 21
2.4 管脚描述 23
2.4.1 SPMC75F2413A 80-Pin QFP 封装 23
2.4.2 SPMC75F2413A 64-Pin QFP 封装 25
2.4.3 SPMC75F2313A 42-Pin SDIP 封装 26
2.4.4 SPMC75F2313A 44-Pin LFQP 封装 27

第 3 章 时钟模块 29

3.1 简介 29
3.2 RC 振荡器 29
3.3 外部时钟输入 29
3.4 本地晶体振荡器 30
3.5 时钟监控 30
3.6 控制寄存器 31
3.7 应用电路 31
3.7.1 本地晶体振荡方式 31
3.7.2 外部时钟输入 32
3.8 设计参考 32

第 4 章 复位 33

4.1 简介 33

4.2 复位方式 33
4.2.1 上电复位 33
4.2.2 外部复位 34
4.2.3 低电压复位 35
4.2.4 看门狗复位 35
4.2.5 非法地址访问复位 36
4.2.6 非法指令复位 36
4.3 复位源列表 37
4.4 控制寄存器 37
4.5 设计参考 39
4.5.1 外部复位 39
4.5.2 复位状态 40
4.6 启动引导序列 40

第 5 章 CPU 41

5.1 简介 41
5.2 CPU 内部寄存器 42
5.2.1 程序计数器 PC 42
5.2.2 堆栈指针 SP 42
5.2.3 状态标志寄存器 SR 42
5.2.4 CPU 状态标志寄存器 FR 43
5.2.5 基址指针寄存器 BP 44
5.2.6 通用寄存器 R1-R4 44
5.2.7 辅助通用寄存器 SR1-SR4 44
5.3 CPU 寻址方式 44
5.3.1 立即数寻址 45
5.3.2 直接寻址 45
5.3.3 寄存器寻址 47
5.3.4 寄存器间接寻址 47
5.3.5 寄存器间接增量寻址 48
5.3.6 基址变址寻址 48
5.3.7 多重间接寻址 49

第 6 章 存储结构 51

6.1 简介 51
6.2 FLASH 51
6.3 SRAM 52
6.4 外设控制寄存器 52

6.5 复位和中断入口地址 55

第 7 章 FLASH 的存储和控制 56

7.1 简介 56
7.2 控制寄存器 56
7.2.1 Flash 访问控制寄存器 56
7.2.2 Flash 控制命令寄存器 57
7.2.3 系统选项寄存器 58
7.3 FLASH 写保护 60

第 8 章 中断 62

8.1 简介 62
8.2 IRQ 中断向量分配 62
8.3 控制寄存器 63
8.4 中断模型 70
8.5 设计参考 75

第 9 章 I/O 端口 78

9.1 简介 78
9.2 I/O 口结构 78
9.2.1 按键唤醒中断 79
9.3 IO 端口 A 80
9.4 IO 端口 B 85
9.5 IO 端口 C 89
9.6 IO 端口 D 93
9.7 I/O 端口编程注意事项 95
9.8 I/O 初始化 95

第 10 章 节电模式和唤醒功能 97

10.1 简介 97
10.2 唤醒源 99
10.3 控制寄存器 99
10.4 设计参考 101

第 11 章 PDC 定时器模块 102

11.1 简介 102
11.2 PDC 定时器的功能 102
11.3 PDC 定时器输入/输出/特殊功能管脚 105
11.4 PDC 定时器的工作模式 105

11.4.1 连续递增计数和边沿 PWM 输出模式 105
11.4.2 连续递增/递减计数和中心 PWM 输出模式 107
11.5 PDC 定时器控制寄存器 108
11.5.1 PDC 定时器的控制寄存器 109
11.5.2 PDC 定时器的输入输出控制寄存器 111
11.5.3 PDC 定时器的中断使能寄存器 112
11.5.4 PDC 定时器的中断状态寄存器 113
11.5.5 定时器启动寄存器 115
11.5.6 PDC 定时器的侦测位置改变控制寄存器 116
11.5.7 PDC 定时器的位置侦测数据寄存器 118
11.5.8 PDC 定时器的计数寄存器 118
11.5.9 PDC 定时器的通用寄存器 119
11.5.10 PDC 定时器的缓冲寄存器 120
11.5.11 PDC 定时器的周期寄存器 120
11.6 PDC 定时器的标准计数操作 121
11.7 PDC 定时器的 PWM 比较匹配输出操作 121
11.8 PDC 定时器的输入捕获操作 122
11.9 PDC 定时器的侦测位置改变(PDC)模式操作 126
11.10 PDC 定时器的相位计数模式操作 129
11.10.1 相位计数模式 1 129
11.10.2 相位计数模式 2 130
11.10.3 相位计数模式 3 130
11.10.4 相位计数模式 4 131
11.11 设计参考 132

第 12 章 TPM 定时器 2 模块 136

12.1 简介 136
12.2 定时器 2 的功能 136
12.3 TPM 定时器 2 的输入/输出/特殊功能引脚 138
12.4 TPM 定时器 2 的计数操作 138
12.4.1 边沿 PWM 输出方式的连续递增计数模式 138
12.4.2 中心 PWM 输出的连续递增/递减计数模式 140
12.5 定时器 2 控制寄存器 142
12.5.1 TPM 定时器 2 控制寄存器 142
12.5.2 TPM 定时器 2 输入输出控制寄存器 143
12.5.3 TPM 定时器 2 的中断使能寄存器 144
12.5.4 TPM 定时器 2 中断状态寄存器 145

12.5.5 定时器计数启动寄存器 146
12.5.6 TPM 定时器 2 计数寄存器 147
12.5.7 TPM 定时器 2 的通用寄存器 147
12.5.8 TPM 定时器 2 的缓冲寄存器 149
12.5.9 TPM 定时器 2 周期寄存器 149
12.6 TPM 定时器 2 的操作 150
12.6.1 标准计数操作 150
12.6.2 PWM 比较匹配输出操作 151
12.6.3 输入捕获操作 152
12.7 设计参考 154

第 13 章 MCP 定时器模块 157

13.1 简介 157
13.2 MCP 定时器 3 和 MCP 定时器 4 的特性 157
13.3 MCP 定时器 3 和定时器 4 输入/输出/特殊功能引脚 160
13.4 MCP 定时器的计数操作 160
13.4.1 边沿 PWM 的连续递增计数模式 160
13.4.2 中心 PWM 的连续递增/递减计数模式 162
13.5 MCP 定时器控制寄存器 162
13.5.1 MCP 定时器的控制寄存器 164
13.5.2 MCP 定时器的输出控制寄存器 165
13.5.3 MCP 定时器的中断使能寄存器 166
13.5.4 MCP 定时器的中断状态寄存器 167
13.5.5 MCP 定时器的输出使能寄存器 168
13.5.6 MCP 定时器的输出控制寄存器 169
13.5.7 MCP 定时器的死区及其控制寄存器 173
13.5.8 MCP 定时器的错误输入控制寄存器 174
13.5.9 MCP 定时器的错误释放寄存器 178
13.5.10 MCP 定时器的过载保护控制和状态寄存器 179
13.5.11 MCP 定时器的 PWM 模块写使能控制寄存器 183
13.5.12 MCP 定时器完整载入寄存器 184
13.5.13 定时器启动寄存器 185
13.5.14 MCP 定时器的计数寄存器 186
13.5.15 MCP 定时器的通用寄存器 186
13.5.16 MCP 定时器的缓冲寄存器 187
13.5.17 MCP 定时器的周期寄存器 188
13.6 MCP 定时器的操作 188

13.6.1 标准计数的操作 188
13.6.2 PWM 输出操作 189
13.7 设计参考 190

第 14 章 比较匹配定时器 201

14.1 简介 201
14.2 控制寄存器 201
14.2.1 比较匹配定时器起动寄存器 202
14.2.2 比较匹配定时器控制和状态寄存器 202
14.2.3 比较匹配定时器的计数器 204
14.2.4 比较匹配定时器的周期寄存器 204
14.3 设计参考 205

第 15 章 时基模块和蜂鸣器模块 206

15.1 简介 206
15.2 控制寄存器 206
15.2.1 时基复位寄存器 206
15.2.2 蜂鸣器输出控制寄存器 207
15.3 设计参考 207

第 16 章 SPI 标准外设接口 208

16.1 简介 208
16.2 SPI 控制引脚配置 208
16.3 控制寄存器 209
16.3.1 SPI 控制寄存器 209
16.3.2 SPI 发送状态寄存器 210
16.3.3 SPI 发送缓冲寄存器 211
16.3.4 SPI 接收状态寄存器 211
16.3.5 SPI 接收缓冲寄存器 212
16.4 SPI 运行模式 212
16.4.1 主模式 213
16.4.2 从模式 213
16.5 设计参考 214

第 17 章 通用异步串行通讯-UART 216

17.1 简介 216
17.1.1 UART 帧结构 216
17.1.2 UART 引脚配置 216

17.2 控制寄存器 217
17.2.1 UART 数据寄存器 217
17.2.2 UART 错误状态寄存器 218
17.2.3 UART 控制寄存器 219
17.2.4 UART 波特率设定寄存器 220
17.2.5 UART 状态寄存器 221
17.3 UART 的操作 222
17.3.1 波特率设置 222
17.3.2 数据发送 222
17.3.3 数据接收 223
17.4 设计参考 224

第 18 章 10 位模/数转换器 226

18.1 简介 226
18.2 控制寄存器 226
18.2.1 ADC 设置寄存器 226
18.2.2 ADC 控制寄存器 227
18.2.3 ADC 输入通道使能寄存器 228
18.2.4 ADC 数据寄存器 229
18.3 ADC 转换时序 231
18.4 使用注意事项 232
18.5 设计参考 232

第 19 章 看门狗 234

19.1 简介 234
19.2 控制寄存器 234
19.2.1 看门狗控制寄存器 234
19.2.2 看门狗清除寄存器 235
19.3 设计参考 236

第 20 章 指令集 237

20.1 简介 237
20.2 算术逻辑单元(ALU)操作 238
20.3 条件分支跳转 239
20.4 指令集 240
20.4.1 段长调用 240
20.4.2 段间长调用指令 241
20.4.3 长跳转 241

20.4.4 间接长跳转 242
20.4.5 RETF 子程序返回 243
20.4.6 RETI 中断返回 243
20.4.7 Break 软件中断 244
20.4.8 FIR_MOV 开/关 244
20.4.9 第二寄存器组选取指令 245
20.4.10 Fraction 开/关模式 246
20.4.11 IRQ 嵌套模式使能或禁止指令 246
20.4.12 IRQ 开/关 247
20.4.13 FIQ 开/关 248
20.4.14 中断设置 248
20.4.15 DS 段地址直接访问指令 248
20.4.16 数据段段地址设置指令(6 位短立即数) 249
20.4.17 空操作 249
20.4.18 处理器标志寄存器存取指令 250
20.4.19 除法指令 DIVS/DIVQ 250
20.4.20 EXP 指令 252
20.4.21 位操作 253
20.4.22 移位操作指令 254
20.4.23 寄存器乘法指令(Mul) 255
20.4.24 乘法累加运算(Muls) 256
20.4.25 条件分支跳转(分支) 257
20.4.26 压栈和出栈操作(通过间接寻址) 258
20.4.27 基址变址寻址方式的 ALU 操作 (BP+6 位短立即数) 258
20.4.28 6 位立即数寻址的 ALU 操作 (Imm6) 259
20.4.29 寄存器间接寻址和寄存器间接变址寻址的 ALU 操作 260
20.4.30 16 位立即数寻址的 ALU 操作 261
20.4.31 16 位直接地址寻址的 ALU 操作 262
20.4.32 6 位直接地址寻址的 ALU 操作 262
20.4.33 寄存器寻址的 ALU 操作 263

第 21 章 硬件概述 265

21.1 系统控制寄存器 265
21.2 看门狗控制寄存器 265
21.3 节电模式控制寄存器 266
21.4 I/O 端口的控制寄存器 267
21.5 中断模块控制寄存器 273

21.6 通用定时器模块控制寄存器 274
21.7 CMT 定时器控制寄存器 283
21.8 TMB/蜂鸣器控制寄存器 284
21.9 SPI 模块控制寄存器 285
21.10 UART 模块控制寄存器 286
21.11 ADC 模块控制寄存器 288
21.12 FLASH 模块控制寄存器 289

第 22 章 SPMC75X 系列微控制器开发系统 290

22.1 开发系统设置示意图 290
22.2 SPMC75X 系列微控制器 EVM 板 V1.1 290
22.3 SPMC75X 系列微控制器的应用电路 291
22.3.1 SPMC75F2413A QFP80 引脚应用 291
22.3.2 SPMC75F2413A QFP64 引脚应用 292
22.4 SPMC75F2413A EVM 板 V1.1 示意图 293

第 23 章 凌阳创新 U'NSP 集成开发环境(IDE) 297

23.1 简介 297
23.2 安装 297
23.3 快速启动 298
23.4 μ'NSP® IDE 介绍 298
23.4.1 主窗口 298
23.4.2 工作区(Workspace)窗口 298
23.4.3 输出（Output）窗口 299
23.4.4 编辑（Edit）窗口 299
23.5 工程 309
23.5.1 工作区窗口 309
23.6 源代码 326
23.6.1 创建文档 326
23.6.2 打开文档 327
23.6.3 保存文档 327
23.6.4 书签 327
23.6.5 字符串查找 328
23.6.6 二进制编辑 330
23.7 资源 330
23.7.1 资源 ID 330
23.7.2 添加资源 330
23.7.3 改变工程内的资源 331

23.8 调试器 331
23.8.1 运行控制 331
23.8.2 调试窗口 332
23.9 代码分析器 333
23.9.1 代码分析器功能 333
23.9.2 代码分析器操作 333
23.10 工具栏 334
23.10.1 定制用户开发环境 334
23.10.2 自定义编辑器 338
23.11 其他工具 341
23.11.1 Dump File 341
23.11.2 LibMaker 342
23.11.3 MemoryMap 342
23.12 工具栏按钮 343

第 24 章 凌阳创新 DMC 工具介绍 345

24.1 DMC 简介 345
24.2 系统要求 345
24.3 安装指南 345
24.4 DMC 工具组成 346
24.5 监控窗口 348
24.6 图形窗口 350
24.7 DMC 应用举例 351
24.7.1 启动 DMC 工具 351
24.7.2 系统监控 353
24.8 DMC 操作 354
24.9 技术支持 358

第 25 章 附录 A 保留字 359

修订记录

日期	版本	编写及修订者	编写及修订说明
2005/11/14	1.7	牟联树	英文版移除 preliminary, 同步更新
2005/09/21	1.6	牟联树	1. 修正 SPMC75x 编程指南 1.1 版中的错误，并根据最新的英文版（1.6 版）对之前的一些描述作了更改。 2. 更改 P_POS0_DectCtrl 和 P_POS1_DectCtrl 寄存器的功能描述。 3. 这版的改动比较大，包括结构和内容上都做了很多更改。因此这个版本和上一个版本很是不同。主要增加了对 IDE 和 DMC 工具使用说明。同时，为了符合大陆的描述习惯，这个中文版本的结构和最新的英文版也会有不同。
2005/01/03	1.1	叶新华	1. 修改振荡器应用电路，加入图 3-3，3-4； 2. 修改 2.9 节芯片管脚描述中 XTAL2 作用； 3. ADC 章加入使用注意事项； 4. 修改 11、12 章 Captue 操作,加入表格； 5. P_OIx_Ctrl 的 RTOL 位功能描述改为自由； 6. 修改第 15 章蜂鸣器模块输出频率表,精确到小数.

第1章 概述

1.1 简介

SPMC75X 系列微控制器是由凌阳创新科技公司设计开发的工业级的 16 位微控制器芯片，其核心微处理器μ'nSP™（发音为 micro-n-SP）集成了多功能 I/O 口、同步和异步串行口、ADC、定时计数器等功能模块，以及多功能捕获比较模块、BLDC 电机驱动专用位置侦测接口、两相增量编码器接口、能产生各种电机驱动波形的 PWM 发生器等特殊硬件模块。利用这些硬件模块支持，SPMC75X 系列微控制器可以完成诸如家电用变频驱动器、标准工业变频驱动器、变频电源、多环伺服驱动系统等复杂应用。

1.2 芯片特性

- 高性能的 16 位内核
  - 凌阳创新 16 位 u'nSP 处理器(ISA 1.2)
  - 2 种低功耗模式：Wait/Standby
  - 片内低电压检测电路
  - 片内基于锁相环的时钟发生模块
  - 最高运行速度：24MHz
- 芯片内存储器
  - 32KW (32K×16) Flash
  - 2KW (2K×16) SRAM
- 工作温度： -40 ℃～85 ℃
- 10 位的 ADC 模块
  - 可编程的换速率，最大转换速率 100Ksps
  - 8 个外部输入通道
  - 可与 PDC 或是 MCP 等定时器联动，实现电机控制中的电参量测量
- 串行通讯接口
  - 通用异步串行通讯接口(UART)
  - 标准外围接口(SPI)
- 最多 64 个通用输入输出管脚
- 可编程看门狗定时器
- 内嵌在线仿真功能，可实现的在线仿真、调试和下载
- PDC 定时器
  - 两个 PDC 定时器:PDC0 和 PDC1
  - 可同时处理三路捕获输入
  - 可产生三路 PWM 输出（中心对称或边沿方式）
  - BLDC 驱动的专用位置侦测接口
  - 两相增量码盘接口，支持四种工作模式，拥有四倍频电路
  - 普通的定时器功能
- MCP 定时器
  - 两个 MCP 定时器:MCP3 和 MCP4
  - 能够产生三相六路可编程的 PWM 波形（中心对称或边沿方式），如三相的 SPWM、SVPWM 等
  - 提供 PWM 占空比值同步载入逻辑
  - 可选择与 PDC 的位置侦测变化同步
  - 可编程的硬件死区插入功能，死区时间可设定
  - 可编程的错误和过载保护逻辑
  - 普通的定时器功能
- TPM 定时器 2
  - 可同时处理二路捕获输入
  - 可产生二路 PWM 输出（中心对称或边沿方式）
  - 普通的定时器功能
- 两个 CMT 定时器
  - 通用 16 位定时器

1.3 应用领域

- 变频家电
- 工业变频器
- 变频电源
- 不间断电源 UPS
- 风扇控制

第2章　结构概述

2.1 简介

SPMC75X 系列微控制器使用 RISC 架构，程序空间（Flash）和数据空间（SRAM）统一编址。其结构分为 CPU 内核、外围功能模块二部分。其中内核是芯片的基本部分，如 CPU、存储器等；外围则包括定时/计数器、时钟系统、ADC 模块、UART 通信模块等。SPMC75X 系列微控制器结构如图 2-1 和 2-2 所示。

2.2 芯片结构

2.2.1 SPMC75F2413A 内部结构图

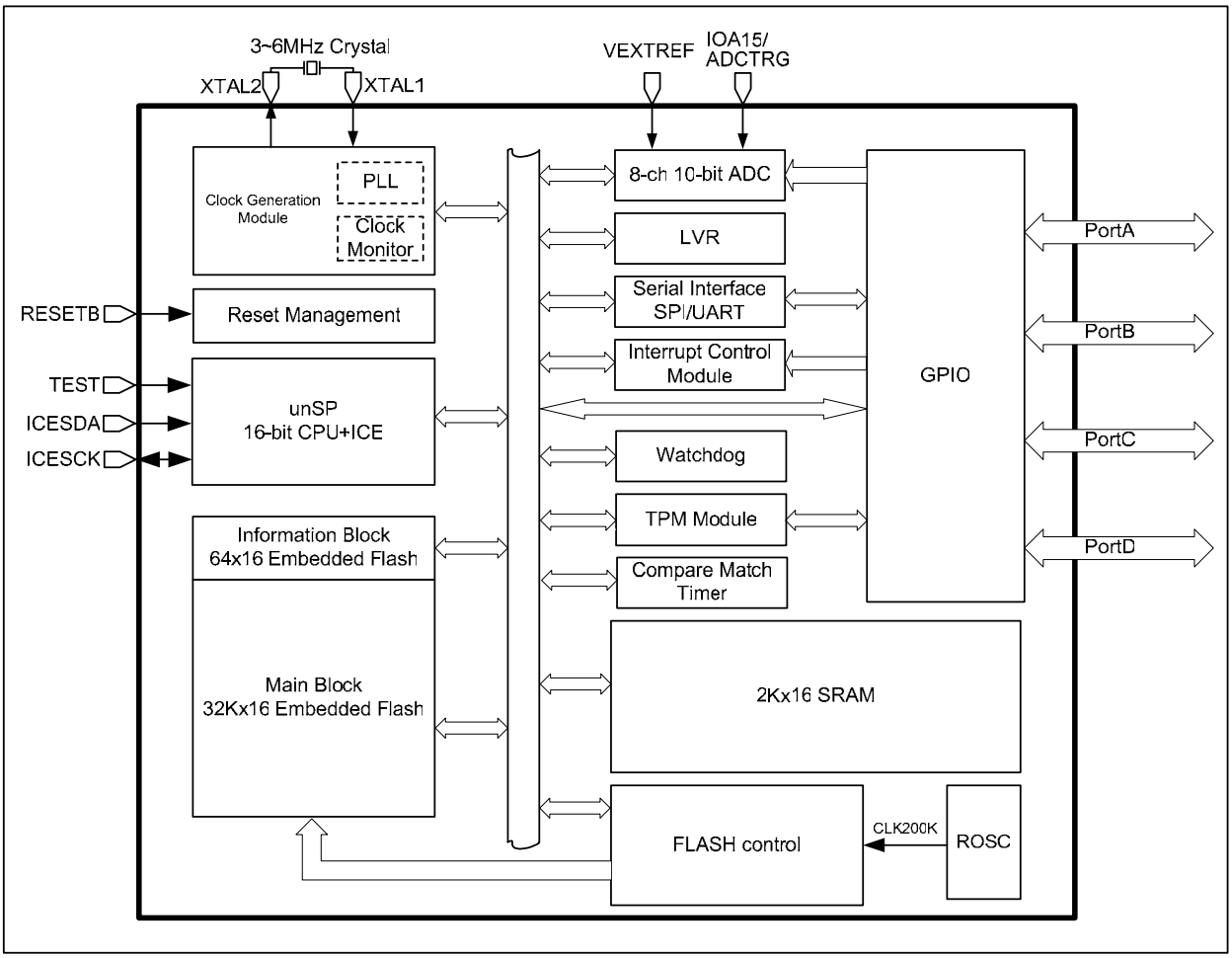


图 2-1　SPMC75F2413A 功能框图

2.2.2 SPMC75F2313A 内部结构图

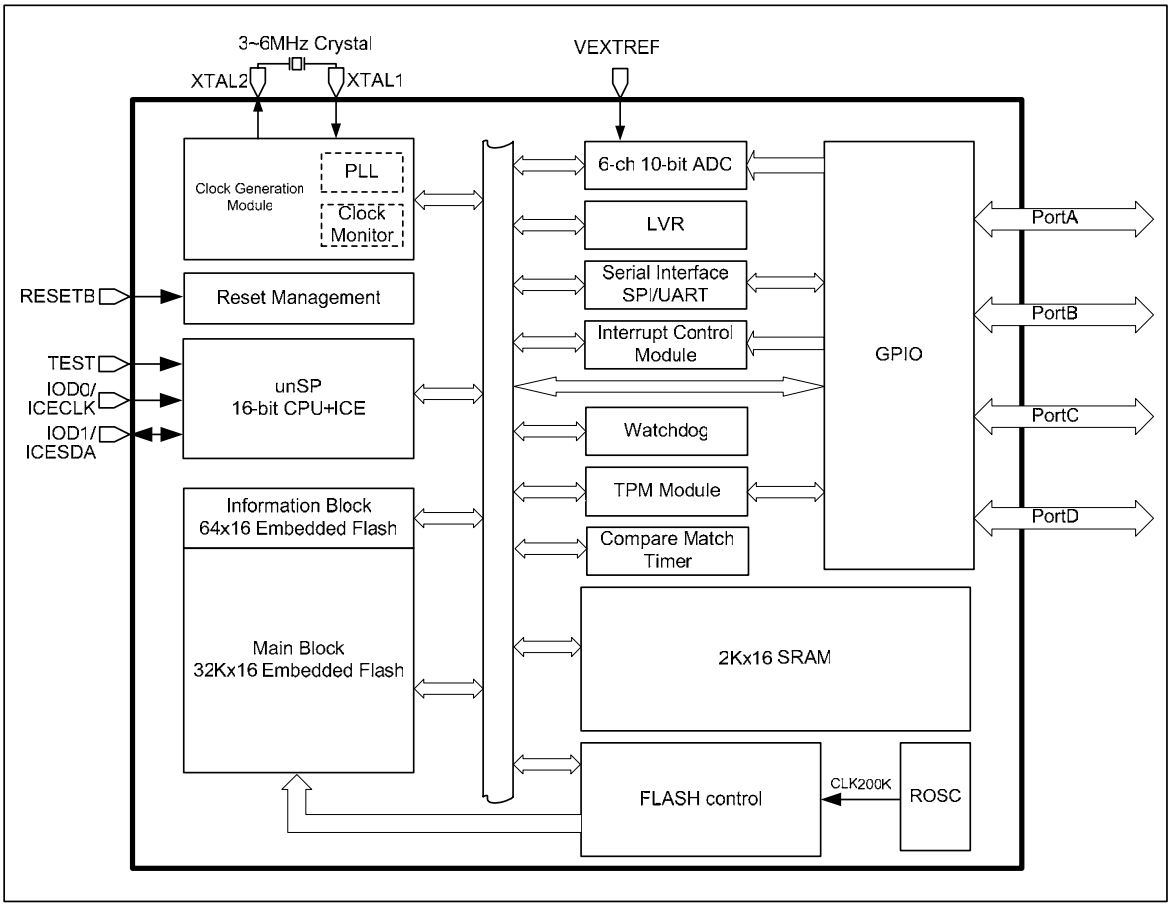


图 2-2 SPMC75F2313A 功能框图

2.2.3 硬设资源表

SPMC75X 系列微控制器现共有两颗芯片：SPMC75F2413A 和 SPMC75F2313A，每颗集成了不同的外设功能模块。详见表 2-1 所述：

表 2-1SPMC75X 系列微控制器的功能表

型号 / 功能		SPMC75F2313A	SPMC75F2413A
内部存储器	程序区 (Flash)	32K words	32K words
	数据区 (SRAM)	2K words	2K words
工作时钟		12 - 24 MHz	12 - 24 MHz
工作电压		4.5 - 5.5V	4.5 - 5.5V
输入/输出口 (最多可达)		33	64
定时器	PDC 定时器	16 bit x 2	16 bit x 2
	TPM 定时器	16 bit x 1	16 bit x 1
	MCP 定时器	16 bit x 1	16 bit x 2
	CMT 定时器	16 bit x 2	16 bit x 2
通用 PWM 输出		16 bit x 8	16 bit x 8

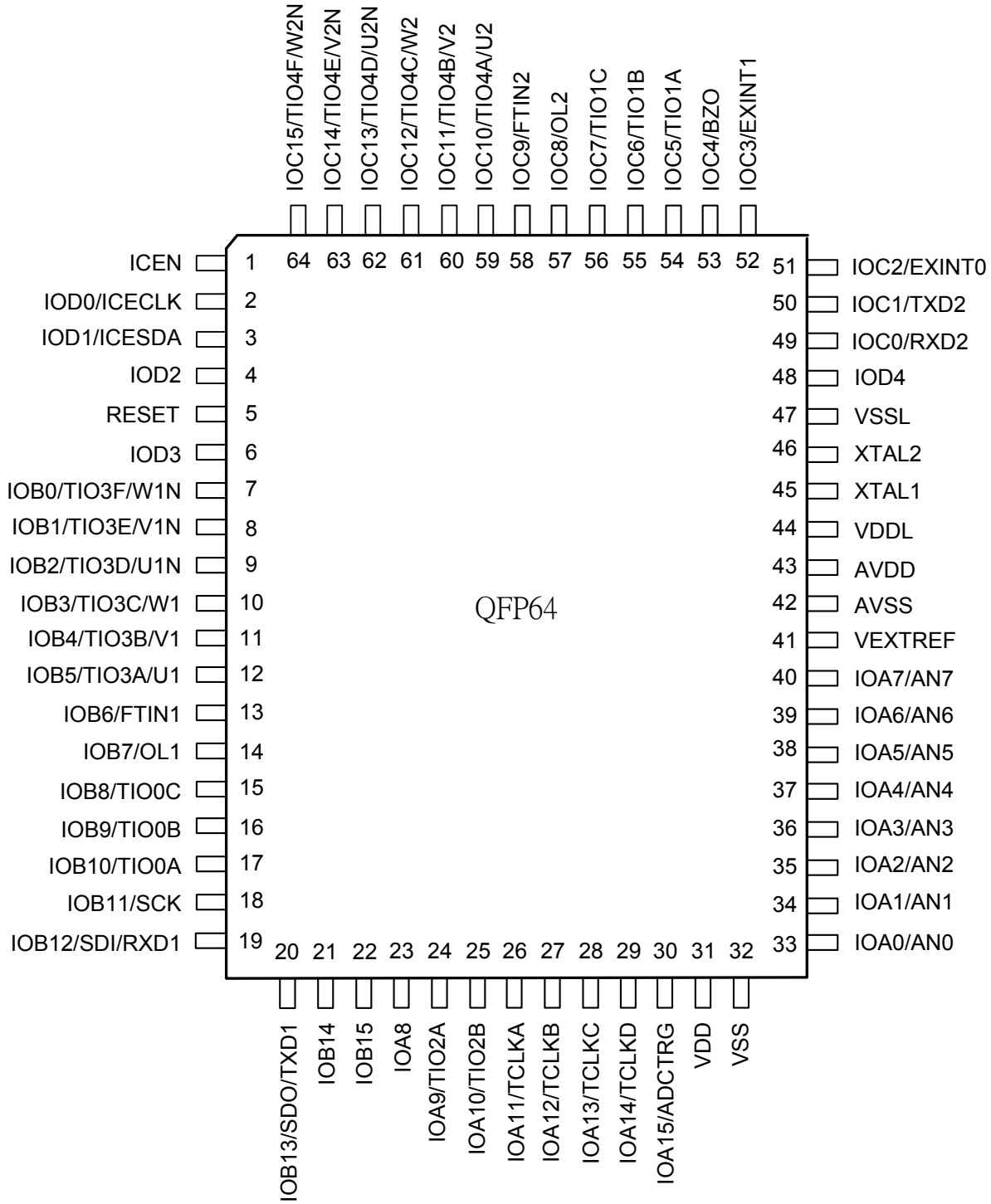
型号 / 功能	SPMC75F2313A	SPMC75F2413A
用于电机控制的 PWM 输出	16 bit x 6	16 bit x 12
捕获输入	8	8
位置检测	PDC 1 定时器	PDC0 , PDC 1 定时器
相位计数模式	PDC 1 定时器	PDC0 , PDC 1 定时器
模/数转换	10 位 6 通道	10 位 8 通道
SPI	有	有
UART	300 - 115200 波特率	300 - 115200 波特率
看门狗定时器	5.46 - 699.05 ms	5.46 - 699.05 ms
蜂鸣器输出	没	1 通道 1.465 - 11.718 KHz
外部中断	没	2 通道
封装	42 pin/SDIP 44 pin/LQFP	64 pin/QFP 80 pin/QFP

2.3 芯片封装

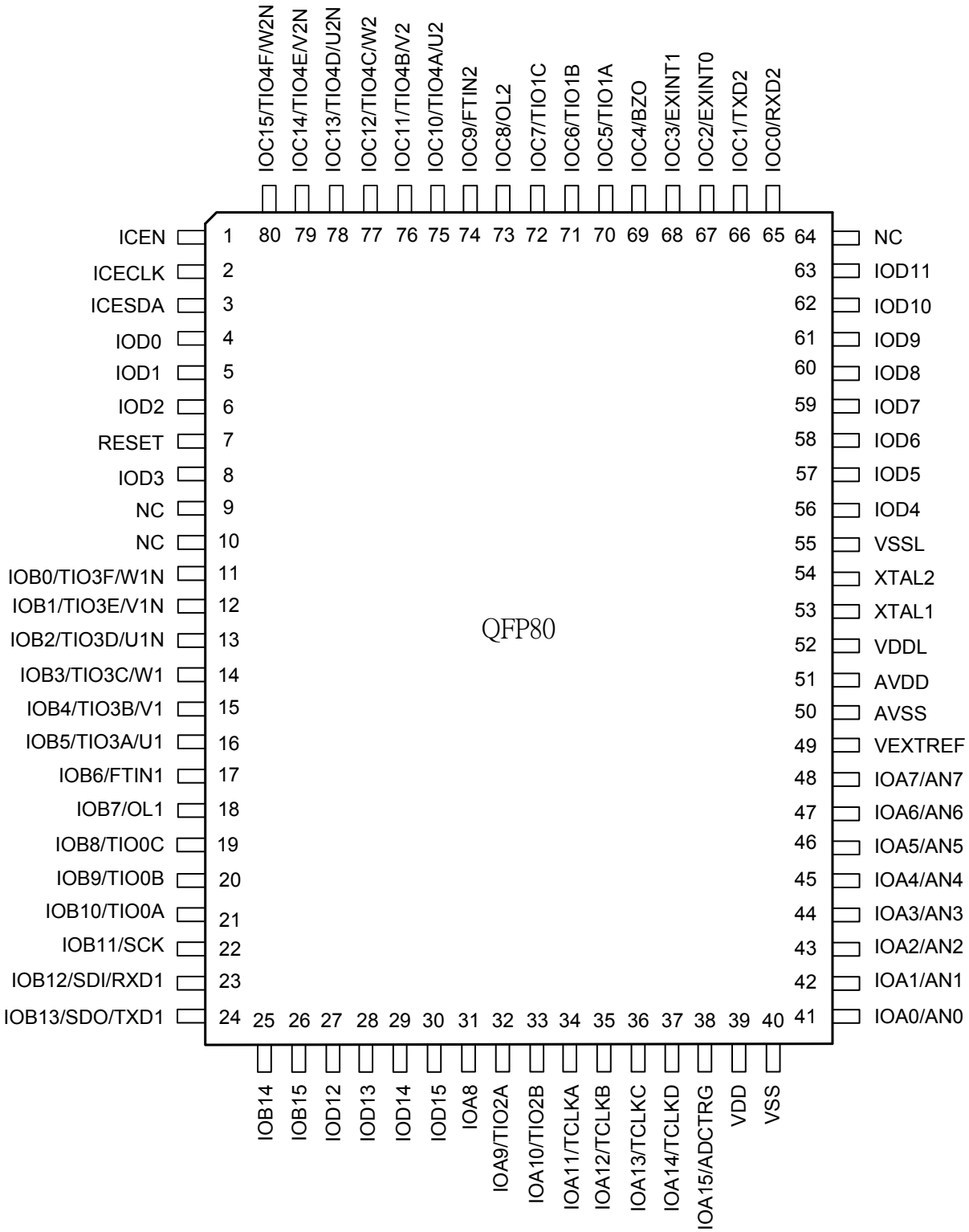
2.3.1 SPMC75F2413A

SPMC75F2413A 的封装方式有： QFP64、QFP80

A：QFP64 – 14 x 20,管脚间距 1.0 mm → 封装号 PQ04


IOC15/TIO4F/W2N
IOC14/TIO4E/V2N
IOC13/TIO4D/U2N
IOC12/TIO4C/W2
IOC11/TIO4B/V2
IOC10/TIO4A/U2
IOC9/FTIN2
IOC8/OL2
IOC7/TIO1C
IOC6/TIO1B
IOC5/TIO1A
IOC4/BZO
IOC3/EXINT1
ICEN 1
IOD0/ICECLK 2
IOD1/ICESDA 3
IOD2 4
RESET 5
IOD3 6
IOB0/TIO3F/W1N 7
IOB1/TIO3E/V1N 8
IOB2/TIO3D/U1N 9
IOB3/TIO3C/W1 10
IOB4/TIO3B/V1 11
IOB5/TIO3A/U1 12
IOB6/FTIN1 13
IOB7/OL1 14
IOB8/TIO0C 15
IOB9/TIO0B 16
IOB10/TIO0A 17
IOB11/SCK 18
IOB12/SDI/RXD1 19
64 63 62 61 60 59 58 57 56 55 54 53 52
51 IOC2/EXINT0
50 IOC1/TXD2
49 IOC0/RXD2
48 IOD4
47 VSSL
46 XTAL2
45 XTAL1
44 VDDL
43 AVDD
42 AVSS
41 VEXTREF
40 IOA7/AN7
39 IOA6/AN6
38 IOA5/AN5
37 IOA4/AN4
36 IOA3/AN3
35 IOA2/AN2
34 IOA1/AN1
33 IOA0/AN0
20 21 22 23 24 25 26 27 28 29 30 31 32
IOB13/SDO/TXD1
IOB14
IOB15
IOA8
IOA9/TIO2A
IOA10/TIO2B
IOA11/TCLKA
IOA12/TCLKB
IOA13/TCLKC
IOA14/TCLKD
IOA15/ADCTRG
VDD
VSS
QFP64

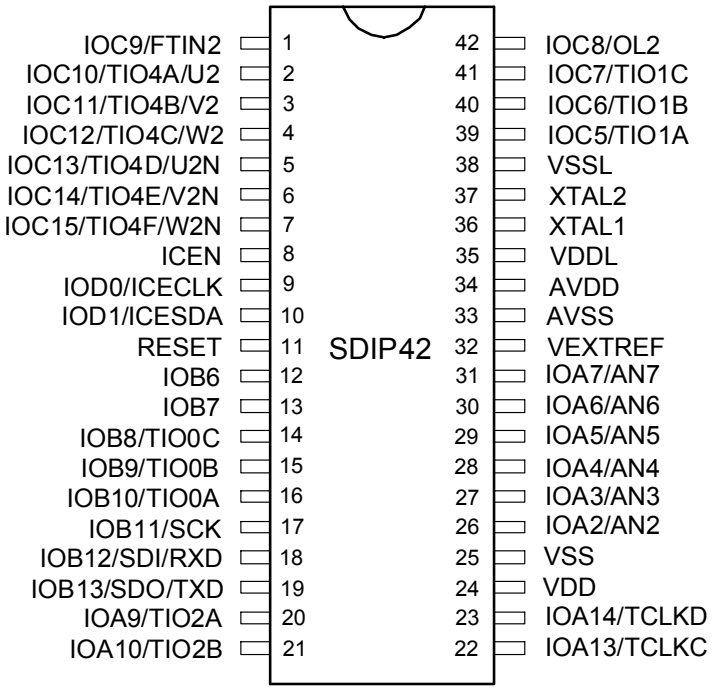
B：QFP80 – 14 x 20 ， 管脚间距 0.8 mm →封装号 PQ05



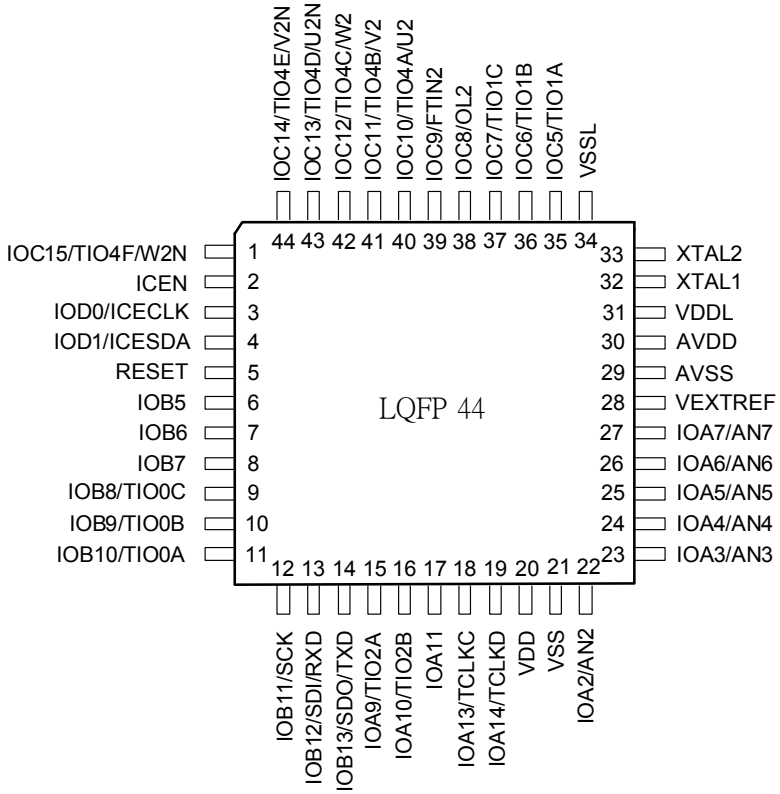
2.3.2 SPMC75F2313A

SPMC75F2313A 的封装方式有：SDIP42、LQFP44

A: SDIP42 –管脚间距 1.78mm →封装号 PD13



B: LQFP44 – 10 x 10, 管脚间距 0.8 mm →封装号 L01


IOC14/TIO4E/V2N
IOC13/TIO4D/U2N
IOC12/TIO4C/W2
IOC11/TIO4B/V2
IOC10/TIO4A/U2
IOC9/FTIN2
IOC8/OL2
IOC7/TIO1C
IOC6/TIO1B
IOC5/TIO1A
VSSL
44 43 42 41 40 39 38 37 36 35 34
IOC15/TIO4F/W2N 1
ICEN 2
IOD0/ICECLK 3
IOD1/ICESDA 4
RESET 5
IOB5 6
IOB6 7
IOB7 8
IOB8/TIO0C 9
IOB9/TIO0B 10
IOB10/TIO0A 11
33 XTAL2
32 XTAL1
31 VDDL
30 AVDD
29 AVSS
28 VEXTREF
27 IOA7/AN7
26 IOA6/AN6
25 IOA5/AN5
24 IOA4/AN4
23 IOA3/AN3
LQFP 44
12 13 14 15 16 17 18 19 20 21 22
IOB11/SCK
IOB12/SDI/RXD
IOB13/SDO/TXD
IOA9/TIO2A
IOA10/TIO2B
IOA11
IOA13/TCLKC
IOA14/TCLKD
VDD
VSS
IOA2/AN2

2.4 管脚描述

类型：I = 输入、 O = 输出、 I/O = 输入/输出

2.4.1 SPMC75F2413A 80-Pin QFP 封装

管脚名称	管脚号	类型	描述
ICEN	1	I (PL)	ICE 仿真/编程模式使能 0V ：普通模式 3.3V ：在线仿真/编程模式
ICECLK	2	I/O	在线仿真时钟输入(3.3V IO)
ICESDA	3	I/O	在线仿真的地址/数据输入或输出复用脚(3.3V IO)
IOD0	4	I/O	IOD0
IOD1	5	I/O	IOD1
IOD2	6	I/O	IOD2
RESET	7	I (PH)	外部复位脚
IOD3	8	I/O	IOD3
NC	9	-	空
NC	10	-	空
IOB0/TIO3F/W1N	11	I/O	IOB0/MCP 定时器 3 的输入/输出 F 或电机驱动 W1N 相输出
IOB1/TIO3E/V1N	12	I/O	IOB1/MCP 定时器 3 的输入/输出 E 或电机驱动 V1N 相输出
IOB2/TIO3D/U1N	13	I/O	IOB2/MCP 定时器 3 的输入/输出 D 或电机驱动 U1N 相输出
IOB3/TIO3C/W1	14	I/O	IOB3/MCP 定时器 3 的输入/输出 C 或电机驱动 W1 相输出
IOB4/TIO3B/V1	15	I/O	IOB4/MCP 定时器 3 的输入/输出 B 或电机驱动 V1 相输出
IOB5/TIO3A/U1	16	I/O	IOB5/MCP 定时器 3 的输入/输出 A 或电机驱动 U1 相输出
IOB6/FTIN1	17	I/O	IOB6/外部故障保护输入脚 1
IOB7/OL1	18	I/O	IOB7/过载保护输入脚 1
IOB8/TIO0C	19	I/O	IOB8 或 PDC 定时器 0 输入/输出 C
IOB9/TIO0B	20	I/O	IOB9 或 PDC 定时器 0 输入/输出 B
IOB10/TIO0A	21	I/O	IOB10 或 PDC 定时器 0 输入/输出 A
IOB11/SCK	22	I/O	IOB11/SPI 时钟输入
IOB12/SDI/RXD1	23	I/O	IOB12/SPI 数据输入/异步通讯串行数据接收口 1
IOB13/SDO/TXD1	24	I/O	IOB13/SPI 数据输出/异步通讯串行数据发送口 1
IOB14	25	I/O	IOB14
IOB15	26	I/O	IOB15
IOD12	27	I/O	IOD12
IOD13	28	I/O	IOD13
IOD14	29	I/O	IOD14
IOD15	30	I/O	IOD15
IOA8	31	I/O	IOA8
IOA9/TIO2A	32	I/O	IOA9/TPM 定时器 2 输入/输出 A
IOA10/TIO2B	33	I/O	IOA10/TPM 定时器 2 输入/输出 B
IOA11/TCLKA	34	I/O	IOA11/外部时钟 A 输入脚
IOA12/TCLKB	35	I/O	IOA11/外部时钟 B 输入脚
IOA13/TCLKC	36	I/O	IOA13/外部时钟 C 输入

管脚名称	管脚号	类型	描述
IOA14/TCLKD	37	I/O	IOA14 外部时钟 D 输入
IOA15/ADCTRG	38	I/O	IOA15/A/D 转换触发输入
VDD	39	I	电源端
VSS	40	I	接地端
IOA0/AN0	41	I/O	IOA0 或模/数转换通道 0 的模拟量输入端
IOA1/AN1	42	I/O	IOA1 或模/数转换通道 1 的模拟量输入端
IOA2/AN2	43	I/O	IOA2 或模/数转换通道 2 的模拟量输入端
IOA3/AN3	44	I/O	IOA3 或模/数转换通道 3 的模拟量输入端
IOA4/AN4	45	I/O	IOA4 或模/数转换通道 4 的模拟量输入端
IOA5/AN5	46	I/O	IOA5 或模/数转换通道 5 的模拟量输入端
IOA6/AN6	47	I/O	IOA6 或模/数转换通道 6 的模拟量输入端
IOA7/AN7	48	I/O	IOA7 或模/数转换通道 7 的模拟量输入端
VEXTREF	49	I	模/数转换参考电源
AVSS	50	I	模拟地
AVDD	51	I	模拟电源
VDDL	52	O	内核电源滤波，外接 10μF-16V 电容
XTAL1	53	I	3-6MHz 晶体输入
XTAL2	54	I/O	3-6MHz 晶体输出/外部 Clock 输入/晶体振荡器输入
VSSL	55	I	内核地
IOD4	56	I/O	IOD4
IOD5	57	I/O	IOD5
IOD6	58	I/O	IOD6
IOD7	59	I/O	IOD7
IOD8	60	I/O	IOD8
IOD9	61	I/O	IOD9
IOD10	62	I/O	IOD10
IOD11	63	I/O	IOD11
NC	64	-	空
IOC0/RXD2	65	I/O	IOC0/异步通讯串行数据接收口 2
IOC1/TXD2	66	I/O	IOC1/异步通讯串行数据发送口 2
IOC2/EXINT0	67	I/O	IOC 2/外部中断输入 0
IOC3/EXINT1	68	I/O	IOC3/外部中断输入 1
IOC4/BZ0	69	I/O	IOC4/蜂鸣器输出
IOC5/TIO1A	70	I/O	IOC5 /PDC 定时器 1 输入/输出 A
IOC6/TIO1B	71	I/O	IOC6/PDC 定时器 1 输入/输出 B
IOC7/TIO1C	72	I/O	IOC7 PDC 定时器 1 输入/输出 C
IOC8/OL2	73	I/O	IOC8/过载保护输入脚 2
IOC9/FTIN2	74	I/O	IOC9/外部故障保护输入脚 2
IOC10/TIO4A/U2	75	I/O	IOC10/MCP 定时器 4 输入/输出 A 或电机驱动 U2 相输出
IOC11/TIO4B/V2	76	I/O	IOC11/MCP 定时器 4 输入/输出 B 或电机驱动 V2 相输出
IOC12/TIO4C/W2	77	I/O	IOC12/MCP 定时器 4 输入/输出 C 或电机驱动 W2 相输出
IOC13/TIO4D/U2N	78	I/O	IOC13/MCP 定时器 4 输入/输出 D 或电机驱动 U2N 相输出
IOC14/TIO4E/V2N	79	I/O	IOC14/MCP 定时器 4 输入/输出 E 或电机驱动 V2N 相输出
IOC15/TIO4F/W2N	80	I/O	IOC15/MCP 定时器 4 输入/输出 F 或电机驱动 W2N 相输出

2.4.2 SPMC75F2413A 64-Pin QFP 封装

管脚名称	管脚号	类型	描述
ICEN	1	I (PL)	ICE 仿真/编程模式使能 0V ：普通模式 3.3V ：在线仿真/编程模式
IOD0/ICECLK	2	I/O	IOD0/在线仿真时钟输入
IOD1/ICESDA	3	I/O	IOD1/在线仿真的地址/数据复用脚
IOD2	4	I/O	IOD2
RESET	5	I (PH)	外部复位脚
IOD3	6	I/O	IOD3
IOB0/TIO3F/W1N	7	I/O	IOB0/MCP 定时器 3 输入/输出 F 或电机驱动 W1N 相输出
IOB1/TIO3E/V1N	8	I/O	IOB1/MCP 定时器 3 输入/输出 E 或电机驱动 V1N 相输出
IOB2/TIO3D/U1N	9	I/O	IOB2/MCP 定时器 3 输入/输出 D 或电机驱动 U1N 相输出
IOB3/TIO3C/W1	10	I/O	IOB3/MCP 定时器 3 输入/输出 C 或电机驱动 W1 相输出
IOB4/TIO3B/V1	11	I/O	IOB4/MCP 定时器 3 输入/输出 B 或电机驱动 V1 相输出
IOB5/TIO3A/U1	12	I/O	IOB5/MCP 定时器 3 输入/输出 A 或电机驱动 U1 相输出
IOB6/FTIN1	13	I/O	IOB6/外部故障保护输入脚 1
IOB7/OL1	14	I/O	IOB7/过载保护输入脚 1
IOB8/TIO0C	15	I/O	IOB8 或 PDC 定时器 0 输入/输出 C
IOB9/TIO0B	16	I/O	IOB9 或 PDC 定时器 0 输入/输出 B
IOB10/TIO0A	17	I/O	IOB10 或 PDC 定时器 0 输入/输出 A
IOB11/SCK	18	I/O	IOB11 或 SPI 时钟输入/输出
IOB12/SDI/RXD1	19	I/O	IOB12/SPI 数据输入/异步通讯串行数据接收口 1
IOB13/SDO/TXD1	20	I/O	IOB13/SPI 数据输出/异步通讯串行数据发送口 1
IOB14	21	I/O	IOB14
IOB15	22	I/O	IOB15
IOA8	23	I/O	IOA8
IOA9/TIO2A	24	I/O	IOA9 或 TPM 定时器 2 输入/输出 A
IOA10/TIO2B	25	I/O	IOA10 或 TPM 定时器 2 输入/输出 B
IOA11/TCLKA	26	I/O	IOA11 或外部时钟 A 输入
IOA12/TCLKB	27	I/O	IOA12 或外部时钟 B 输入
IOA13/TCLKC	28	I/O	IOA13 或外部时钟 C 输入
IOA14/TCLKD	29	I/O	IOA14 或外部时钟 D 输入
IOA15/ADCTRG	30	I/O	IOA15/A/D 转换触发输入
VDD	31	I	电源端
VSS	32	I	接地端
IOA0/AN0	33	I/O	IOA0 或模/数转换通道 0 的模拟量输入端
IOA1/AN1	34	I/O	IOA1 或模/数转换通道 1 的模拟量输入端
IOA2/AN2	35	I/O	IOA2 或模/数转换通道 2 的模拟量输入端
IOA3/AN3	36	I/O	IOA3 或模/数转换通道 3 的模拟量输入端
IOA4/AN4	37	I/O	IOA4 或模/数转换通道 4 的模拟量输入端
IOA5/AN5	38	I/O	IOA5 或模/数转换通道 5 的模拟量输入端

管脚名称	管脚号	类型	描述
IOA6/AN6	39	I/O	IOA6 或模/数转换通道 6 的模拟量输入端
IOA7/AN7	40	I/O	IOA7 或模/数转换通道 7 的模拟量输入端
VEXTREF	41	I	模/数转换参考电源
AVSS	42	I	模拟地
AVDD	43	I	模拟电源
VDDL	44	O	内核电源滤波，外接 10μF-16V 电容
XTAL1	45	I	3-6MHz 晶体输入
XTAL2	46	I/O	3-6MHz 晶体输出/外部 Clock 输入/晶体振荡器输入
VSSL	47	I	内核地
IOD4	48	I/O	IOD4
IOC0/RXD2	49	I/O	IOC0/异步通讯串行数据接收口 2
IOC1/TXD2	50	I/O	IOC1/异步通讯串行数据发送口 2
IOC2/EXINT0	51	I/O	IOC2/外部中断输入 0
IOC3/EXINT1	52	I/O	IOC3/外部中断输入 1
IOC4/BZO	53	I/O	IOC4/蜂鸣器输出
IOC5/TIO1A	54	I/O	IOC5 或 PDC 定时器 1 输入/输出 A
IOC6/TIO1B	55	I/O	IOA6 或 PDC 定时器 1 输入/输出 B
IOC7/TIO1C	56	I/O	IOA7 或 PDC 定时器 1 输入/输出 C
IOC8/OL2	57	I/O	IOC8/过载保护输入脚 2
IOC9/FTIN2	58	I/O	IOC9/外部故障保护输入脚 2
IOC10/TIO4A/U2	59	I/O	IOC10/MCP 定时器 4 输入/输出 A 或电机驱动 U2 相输出
IOC11/TIO4B/V2	60	I/O	IOC11/MCP 定时器 4 输入/输出 B 或电机驱动 V2 相输出
IOC12/TIO4C/W2	61	I/O	IOC12/MCP 定时器 4 输入/输出 C 或电机驱动 W2 相输出
IOC13/TIO4D/U2N	62	I/O	IOC13/MCP 定时器 4 输入/输出 D 或电机驱动 U2N 相输出
IOC14/TIO4E/V2N	63	I/O	IOC14/MCP 定时器 4 输入/输出 E 或电机驱动 V2N 相输出

2.4.3 SPMC75F2313A 42-Pin SDIP 封装

管脚名称	管脚号	类型	描述
IOC9/FTIN2	1	I/O	IOC9/外部故障保护输入脚 2
IOC10/TIO4A/U2	2	I/O	IOC10/MCP 定时器 4 输入/输出 A 或电机驱动 U2 相输出
IOC11/TIO4B/V2	3	I/O	IOC11/MCP 定时器 4 输入/输出 B 或电机驱动 V2 相输出
IOC12/TIO4C/W2	4	I/O	IOC12/MCP 定时器 4 输入/输出 C 或电机驱动 W2 相输出
IOC13/TIO4D/U2N	5	I/O	IOC13/MCP 定时器 4 输入/输出 D 或电机驱动 U2N 相输出
IOC14/TIO4E/V2N	6	I/O	IOC14/MCP 定时器 4 输入/输出 E 或电机驱动 V2N 相输出
IOC15/TIO4F/W2N	7	I/O	IOC15/MCP 定时器 4 输入/输出 F 或电机驱动 W2N 相输出
ICEN	8	I (PL)	ICE 仿真/编程模式使能 0V ：普通模式 3.3V ：在线仿真/编程模式
IOD0/ICECLK	9	I/O	IOD0/在线仿真时钟输入
IOD1/ICESDA	10	I/O	IOD1/或地址/数据输入输出脚
RESET	11	I (PH)	外部复位脚
IOB6	12	I/O	IOB6

管脚名称	管脚号	类型	描述
IOB7	13	I/O	IOB7
IOB8/TIO0C	14	I/O	IOB8 或 PDC 定时器 0 输入/输出 C
IOB9/TIO0B	15	I/O	IOB9 或 PDC 定时器 0 输入/输出 B
IOB10/TIO0A	16	I/O	IOB10 或 PDC 定时器 0 输入/输出 A
IOB11/SCK	17	I/O	IOB11 或 SPI 时钟输入/输出
IOB12/SDI/RXD1	18	I/O	IOB12/SPI 数据输入/异步通讯串行数据接收口 1
IOB13/SDO/TXD1	19	I/O	IOB13/SPI 数据输出/异步通讯串行数据发送口 1
IOA9/TIO2A	20	I/O	IOA9 或 TPM 定时器 2 输入/输出 A
IOA10/TIO2B	21	I/O	IOA10 或 TPM 定时器 2 输入/输出 B
IOA13/TCLKC	22	I/O	IOA13/外部时钟 C 输入
IOA14/TCLKD	23	I/O	IOA14/外部时钟 D 输入
VDD	24	I	电源端
VSS	25	I	接地端
IOA2/AN2	26	I/O	IOA2 或模/数转换通道 2 的模拟量输入端
IOA3/AN3	27	I/O	IOA3 或模/数转换通道 3 的模拟量输入端
IOA4/AN4	28	I/O	IOA4 或模/数转换通道 4 的模拟量输入端
IOA5/AN5	29	I/O	IOA5 或模/数转换通道 5 的模拟量输入端
IOA6/AN6	30	I/O	IOA6 或模/数转换通道 6 的模拟量输入端
IOA7/AN7	31	I/O	IOA7 或模/数转换通道 7 的模拟量输入端
VEXTREF	32	I	模/数转换参考电源
AVSS	33	I	模拟地
AVDD	34	I	模拟电源
VDDL	35	O	外接电容引脚，接 10μF-16V 电容
XTAL1	36	I	3-6MHz 晶体输入
XTAL2	37	I/O	3-6MHz 晶体输出/外部 Clock 输入/晶体振荡器输入
VSSL	38	I	接地端
IOC5/TIO1A	39	I/O	IOC5 或 PDC 定时器 1 输入/输出 A
IOC6/TIO1B	40	I/O	IOC6 或 PDC 定时器 1 输入/输出 B
IOC7/TIO1C	41	I/O	IOC7 或 PDC 定时器 1 输入/输出 C
IOC8/OL2	42	I/O	IOC8 /过载保护输入脚 2

2.4.4 SPMC75F2313A 44-Pin LFQP 封装

管脚名称	管脚号	类型	描述
IOC15/TIO4F/W2N	1	I/O	IOC15/MCP 定时器 4 输入/输出 F 或电机驱动 W2N 相输出
ICEN	2	I (PL)	ICE 仿真/编程模式使能 0V ：普通模式 3.3V ：在线仿真/编程模式
IOD0/ICECLK	3	I/O	IOD0/在线仿真时钟输入
IOD1/ICESDA	4	I/O	IOD1/或地址/数据输入输出脚
RESET	5	I (PH)	外部复位脚
IOB5	6	I/O	IOB5
IOB6	7	I/O	IOB6

管脚名称	管脚号	类型	描述
IOB7	8	I/O	IOB7
IOB8/TI00C	9	I/O	IOB8 或 PDC 定时器 0 输入/输出 C
IOB9/TI00B	10	I/O	IOB9 或 PDC 定时器 0 输入/输出 B
IOB10/TI00A	11	I/O	IOB10 或 PDC 定时器 0 输入/输出 A
IOB11/SCK	12	I/O	IOB11 或 SPI 时钟输入/输出
IOB12/SDI/RXD1	13	I/O	IOB12/SPI 数据输入/异步通讯串行数据接收口 1
IOB13/SDO/TXD1	14	I/O	IOB13/SPI 数据输出/异步通讯串行数据发送口 1
IOA9/TI02A	15	I/O	IOA9 或 TPM 定时器 2 输入/输出 A
IOA10/TI02B	16	I/O	IOA10 或 TPM 定时器 2 输入/输出 B
IOA11	17	I/O	IOA11
IOA13/TCLKC	18	I/O	IOA13/外部时钟 C 输入
IOA14/TCLKD	19	I/O	IOA14/外部时钟 D 输入
VDD	20	I	电源端
VSS	21	I	接地端
IOA2/AN2	22	I/O	IOA2 或模/数转换通道 2 的模拟量输入端
IOA3/AN3	23	I/O	IOA3 或模/数转换通道 3 的模拟量输入端
IOA4/AN4	24	I/O	IOA4 或模/数转换通道 4 的模拟量输入端
IOA5/AN5	25	I/O	IOA5 或模/数转换通道 5 的模拟量输入端
IOA6/AN6	26	I/O	IOA6 或模/数转换通道 6 的模拟量输入端
IOA7/AN7	27	I/O	IOA7 或模/数转换通道 7 的模拟量输入端
VEXTREF	28	I	模/数转换参考电源
AVSS	29	I	模拟地
AVDD	30	I	模拟电源
VDDL	31	O	内核电源滤波，外接 10μF-16V 电容
XTAL1	32	I	3-6MHz 晶体输入
XTAL2	33	I/O	3-6MHz 晶体输出/外部 Clock 输入/晶体振荡器输入
VSSL	34	I	内核地
IOC5/TI01A	35	I/O	IOC5 或 PDC 定时器 1 输入/输出 A
IOC6/TI01B	36	I/O	IOC6 或 PDC 定时器 1 输入/输出 B
IOC7/TI01C	37	I/O	IOC7 或 PDC 定时器 1 输入/输出 C
IOC8/OL2	38	I/O	IOC8/过载保护输入脚 2
IOC9/FTIN2	39	I/O	IOC9/外部故障保护输入脚 2
IOC10/TI04A/U2	40	I/O	IOC10/MCP 定时器 4 输入/输出 A 或电机驱动 U2 相输出
IOC11/TI04B/V2	41	I/O	IOC11/MCP 定时器 4 输入/输出 B 或电机驱动 V2 相输出
IOC12/TI04C/W2	42	I/O	IOC12/MCP 定时器 4 输入/输出 C 或电机驱动 W2 相输出
IOC13/TI04D/U2N	43	I/O	IOC13/MCP 定时器 4 输入/输出 D 或电机驱动 U2N 相输出
IOC14/TI04E/V2N	44	I/O	IOC14/MCP 定时器 4 输入/输出 E 或电机驱动 V2N 相输出

第3章 时钟模块

3.1 简介

SPMC75X 系列单片机的时钟发生模块有两个。一是内部 RC 振荡器产生的 1600Khz 时钟经分频后为系统提供的 200KHz 辅助时钟源；另一个是系统时钟发生模块，包含一个晶体振荡器（外部连接无源石英晶体或陶瓷晶体）和一个四倍频的锁相环（PLL）模块。当使用无源石英晶体或者陶瓷晶体时（频率范围在 3MHz～ 6MHz 之间），晶体振荡器的输出经锁相环(PLL)四倍频后输出，供系统使用。SPMC75X 系列单片机的时钟发生模块还支持直接外部时钟输入方式，这时的时钟不经过锁相环倍频，直接供系统使用。时钟发生模块的结构如图 2-1 所示。

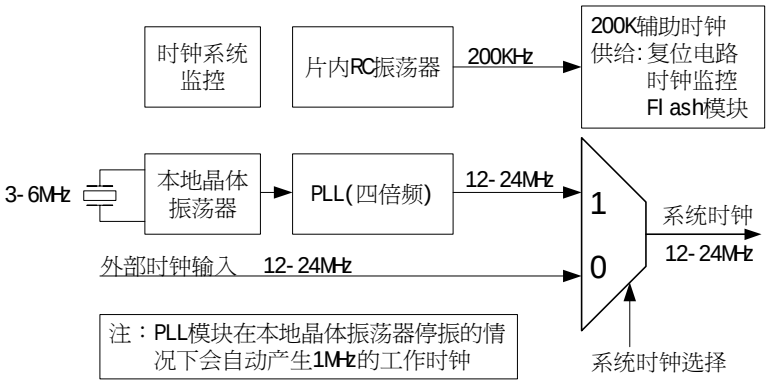


图 2-1 时钟发生模块结构图

3.2 RC 振荡器

1600KHz 的 RC 振荡器为系统提供 200KHz 的辅助时钟源。它为片内 Flash 模块（片内 Flash 模块的擦除和编程均以此时钟为基准）、系统复位模块和时钟失效检测提供基准时钟。为了节电，在 Standby 模式中 RC 振荡器被关闭。

3.3 外部时钟输入

此方式使用外部输入时钟作为系统时钟，如图 2-2 所示。

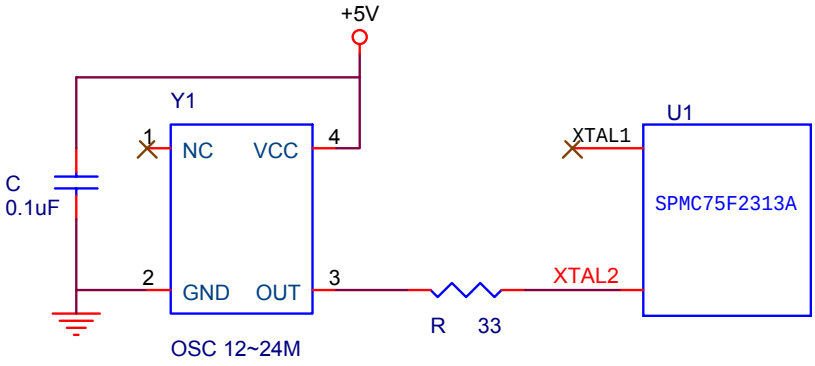


图 2-2 外部时钟输入晶体振荡器

3.4 本地晶体振荡器

此方式使用本地晶体振荡器作为锁相环时钟输入，经锁相环四倍频后的时钟信号为系统时钟。因晶体振荡器和锁相环的稳定需要一定的时间，因此，当上电、系统复位或从 Standby 模式唤醒时，CPU 将等待 82ms(16384 个辅助时钟周期)， 以等待振荡器和锁相环稳定。芯片接晶体振荡器时如图 2-3 所示。在这种方式下，当晶体异常停振时，PLL 模块会提供 1MHz 的应急时钟供系统使用。

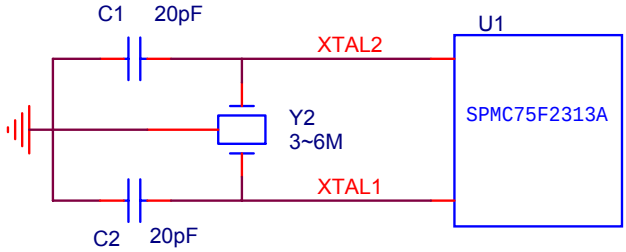


图 2-3 锁相环方式

3.5 时钟监控

时钟监控模块用来监测晶体振荡器是否正常，如果监测到晶体停振，则会将 MCP 定时器 3 和 MCP 定时器 4 的 PWM（TIO3A～F、TIO4A～F）输出置为高阻态（这可以避免在电机驱动应用中，时钟异常对驱动电路造成的损坏），同时会产生相应的中断通知 CPU。监测振荡器时钟停止时序如图 2-4 所示：

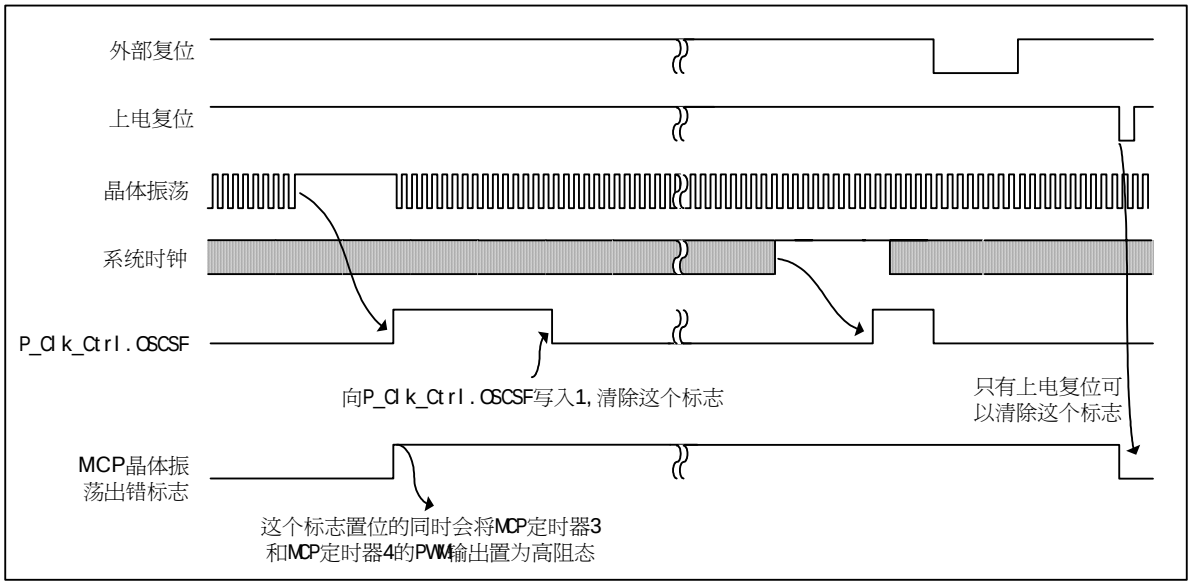


图 2-4 振荡器时钟停止时序图

3.6 控制寄存器

用于监视 CPU 时钟状态。

P_Clk_Ctrl（$7007）：时钟系统控制寄存器

B15	B14	B13	B2	B11	B10	B9	B8
R/W	R/W	R	R	R	R	R	R
0	0	0	0	0	0	0	0
OSCSF	OSCIE	保留					

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

第 15 位　OSCSF：振荡器状态标志。标识芯片内晶振是否正常。如果振荡器停振，此位将被置“1”，向该位写入“1”将清除该标志。

读出

0＝振荡器运行正常

1＝振荡器失效

写入

1＝清除标志

第 14 位　OSCIE：振荡器失效中断使能位

0＝禁止

1＝使能

第 13:0 位　保留

3.7 应用电路

SPMC75X 系列单片机的系统时钟有有两种应用电路：

3.7.1 本地晶体振荡方式

本地晶体振荡方式的应用电路如图 3-5 所示：

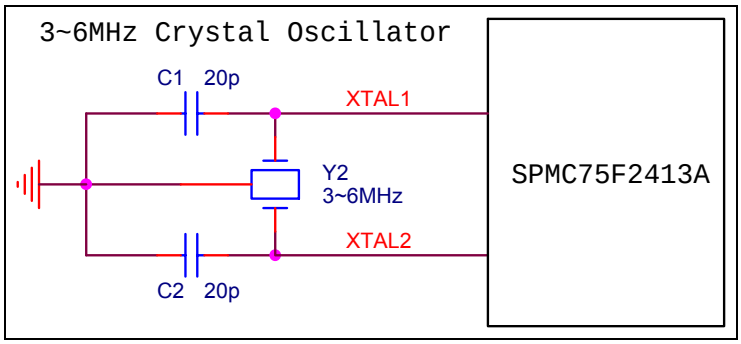


图 3-5 本地晶体振荡方式电路连接

3.7.2 外部时钟输入

外部时钟输入方式的应用电路如图 3-6 所示：

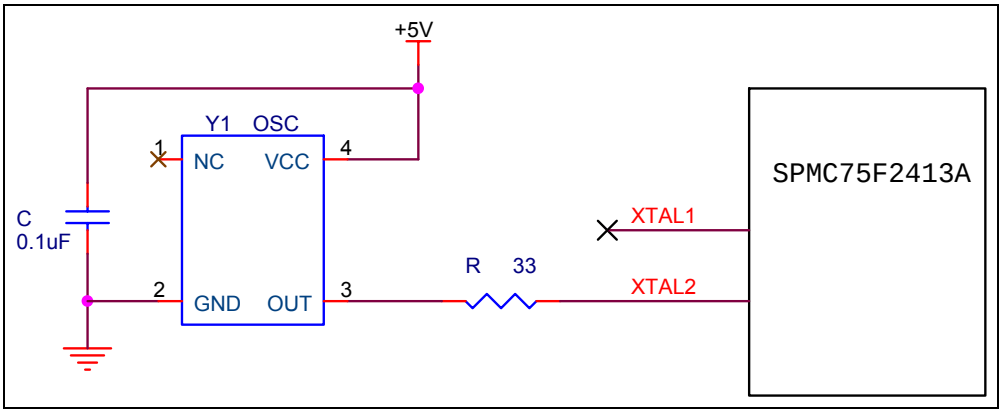


图 3-6 外部时钟输入方式电路连接

3.8 设计参考

```
//使能振荡器失效中断
    P_Clk_Ctrl->B.OSCIE = CB_CLK_OSCIE;        /*使能振荡器中断*/
    if(P_Clk_Ctrl->B.OSCSF & CB_CLK_OSCSF)     /*查询振荡器状态标志*/
    P_Clk_Ctrl->B.OSCSF = CB_CLK_OSCSF;        /* 清除振荡器状态标志 */
```

例 3-1 OSF 中断设计参考

第4章 复位

4.1 简介

SPMC75X 系列微控制器一共有六种复位源。包括外部的上电复位、复位引脚触发和内部复位事件。这六个复位源如下：

1. 上电复位(POR)
2. 外部复位（RESET）
3. 低电压复位(LVR)
4. 看门狗复位(WDTR)
5. 非法地址访问复位(IAR)
   - 访问了非法的地址($0800-$6FFF, $7500-$7FFF)
   - 向 $700C, $700E 写入无效的值
6. 非法指令译码复位(IIS)
   - CPU 得到非法指令译码

4.2 复位方式

4.2.1 上电复位

当系统上电并电源电压上升的斜率大于 0.5 V 每微秒且上升到默认上限值时，上电复位(POR)电路开始工作。上电复位过程将持续 16384 个辅助时钟周期,以等待晶体振荡器和 PLL 稳定。复位周期过后，所有的寄存器重新初始化，时序详见图 4-1。

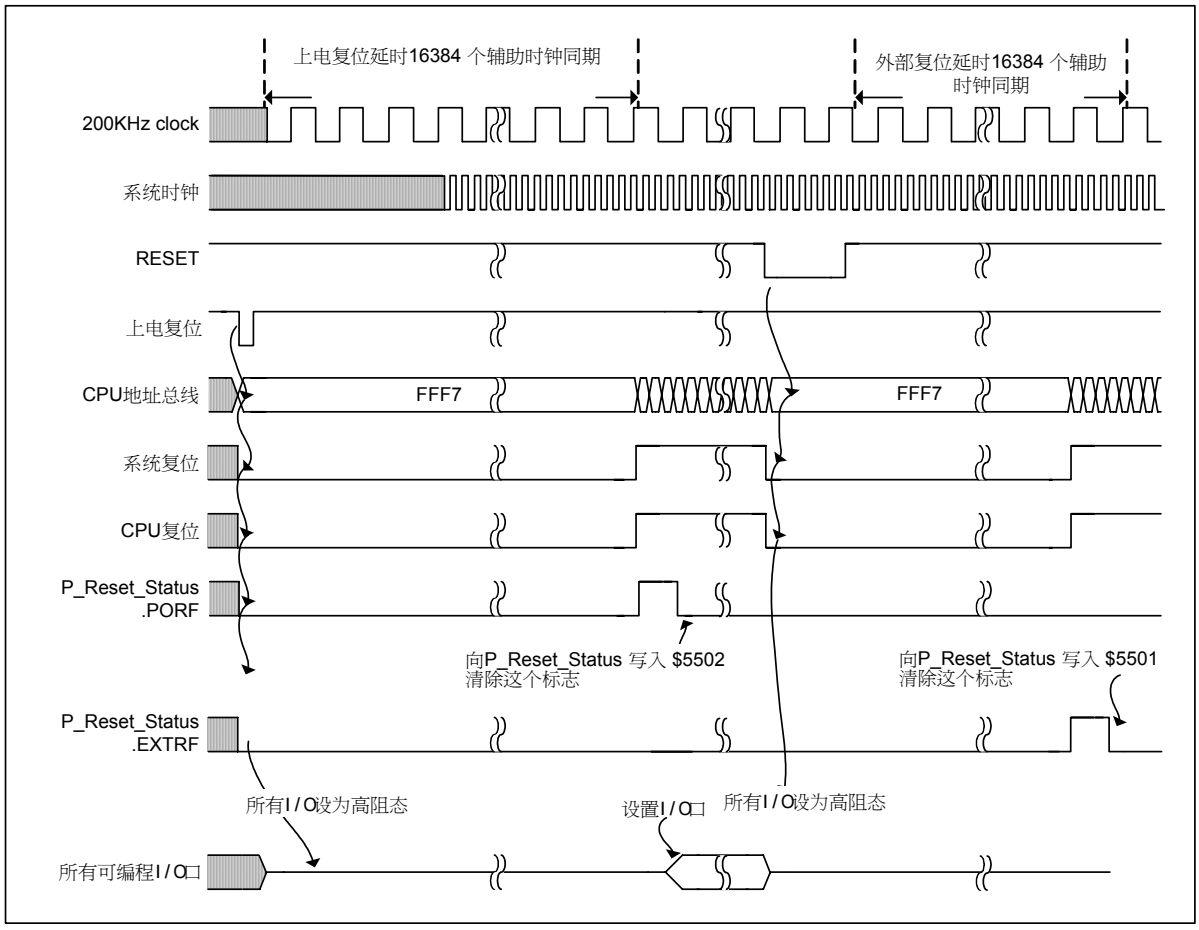


图 4-1 外部复位，上电复位上电时序

4.2.2 外部复位

SPMC75X 系列微控制器提供了外部复位管脚 RESET(低有效)，可进行强制复位。如图 4-2 所示，RESET 连接在一个阻容电路中，外部复位变为无效后内部复位信号将持续 16384 个辅助时钟周期后完成整个复位过程。时序见图 4-1。

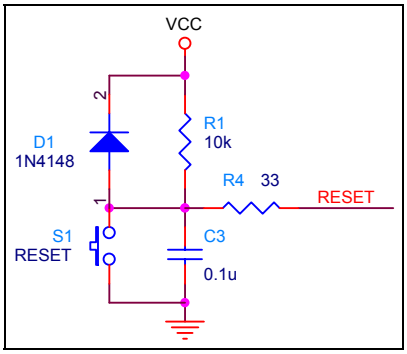


图 4-2 外部复位电路

4.2.3 低电压复位

SPMC75X 系列微控制器提供了低电压复位功能（LVR）。当电源电压降至 4.09V 以下时，低电压复位电路就会使 CPU 及外设复位，当供电电压恢复至 4.19V 以上后，低电压复位信号解除，同时，系统在 16384 个辅助时钟周期后恢复正常运行。图 4-3 描述了典型的低电压复位时序。

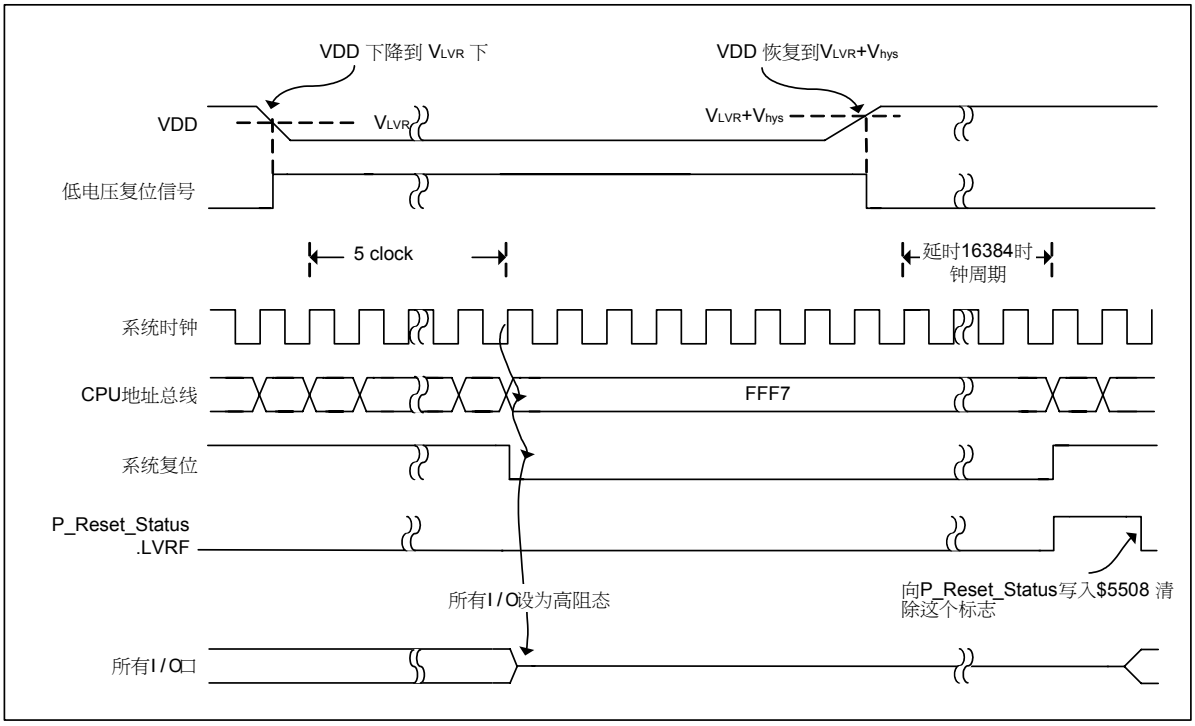


图 4-3 低电压复位时序

4.2.4 看门狗复位

当 SPMC75X 系列微控制器运行到未知的状态下而没有清除看门狗，芯片内看门狗就会使系统进入复位状态。这保证了微控制器不会连续工作在非正常的状态下。通过寄存器 P_System_Option ($8000)B1 位和 P_WatchDog_Ctrl ($700A) B15 位使能看门狗复位功能。用 P_WatchDog_Ctrl 控制寄存器的 B[2:0]设置溢出时间。向 P_WatchDog_Clr(W) ($700B)控制寄存器写入“0xA005”，看门狗计数器就会自动清零并重新计数。如果在看门狗计数器溢出前没有向 P_WatchDog_Clr 控制寄存器写入“0xA005”，看门狗将强制 CPU 或是系统复位。在仿真模式下看门狗复位功能被禁止。看门狗定时器的工作时序如图 4-4 所示。

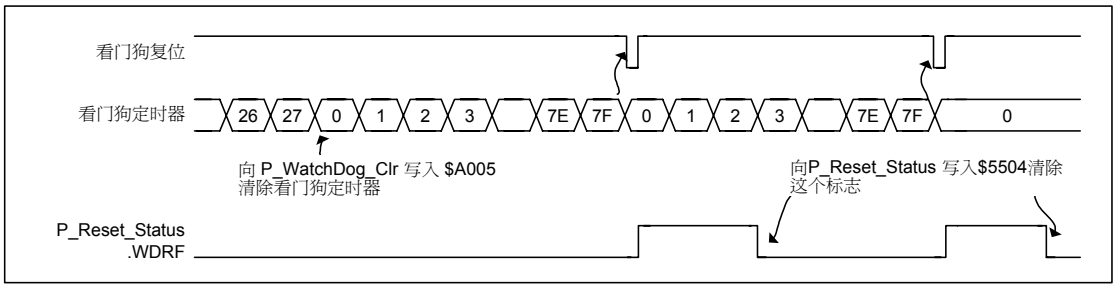


图 4-4 看门狗计数器复位时序

4.2.5 非法地址访问复位

SPMC75X 系列微控制器提供了非法地址访问复位（IAR），一旦访问了非法地址，CPU 将会立即复位。非法的地址范围为$0800～$6FFF，$7500～$7FFF，当指令访问了这些区域系统会立即产生信号复位 CPU。向地址$700C 或$700E 写入了非法数值，也会产生信号导致 CPU 复位。图 4-5 描述了非法的地址复位时序。

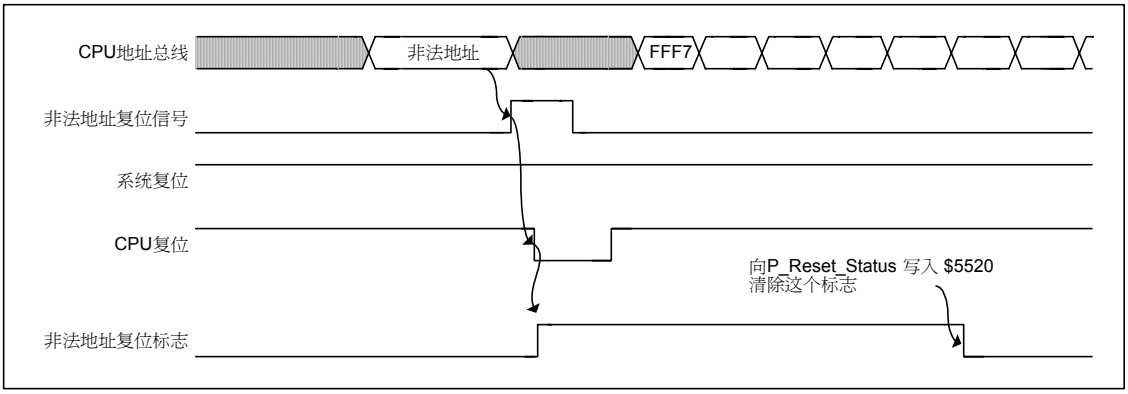


图 4-5 非法的地址复位时序。

4.2.6 非法指令复位

CPU 得到非法指令译码，CPU 会复位（IIR），并设置 P_Reset_Status.IIRF 位。通过这位可以检测系统是否发生过错误指令复位。图 4-6 描述了非法指令复位时序。

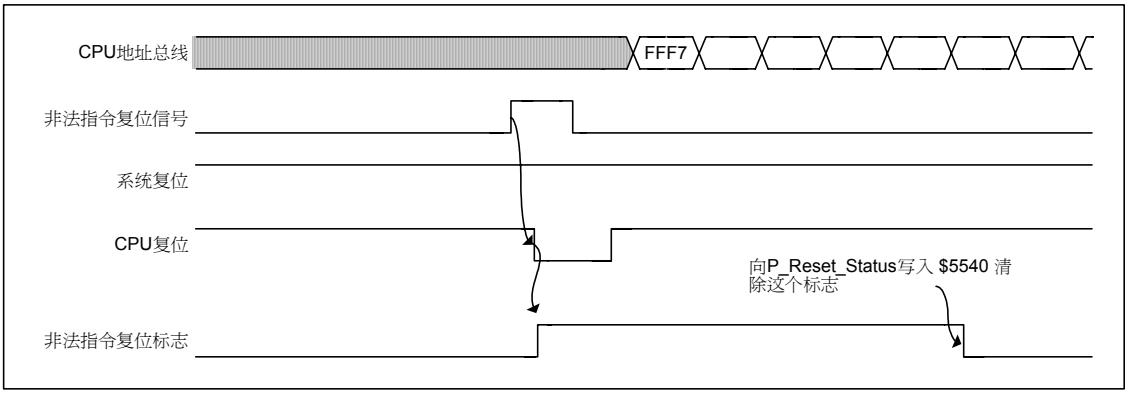


图 4-6 非法指令复位时序

4.3 复位源列表

SPMC75X 系列微控制器复位源和其相应的作用域如下表：

复位源	CPU 内核复位	外设模块复位
外部复位	√	√
上电复位	√	√
看门狗复位	√	可选
低电压复位	√	√
非法地址复位	√	–
非法指令复位	√	–

注：CPU 内核复位是指 CPU 核的复位， CPU 内核复位会使 CPU 核内的所有寄存器初始化为复位默认值；外设模块复位主要是指外设模块（如 ADC 模块、定时器模块）的复位，外设模块复位会使所有外设模块的控制寄存器初始化为复位默认值；复位状态寄存器（P_Reset_Status）会根据复位源初始化相应的复位标志； Flash 控制器只能由上电复位和外部复位来复位的。

4.4 控制寄存器

这个寄存器表示了各种复位状态的标志，可以用软件查询相应的标志位。

P_Reset_Status($7006)：复位状态寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
FCHK							

B7	B6	B5	B4	B3	B2	B1	B0
R	R/W	R/W	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
保留	IIRF	IARF	保留	LVRF	WDRF	PORF	EXTRF

第 15:8 位 FCHK:标志清除校验位

为了正确的清除复位标志，必须向这些位写入 “0x55”。否则，复位标志将不会被清除，这些位读出的数据是“0”。

第 7 位 保留

第 6 位 IIRF: 错误指令复位标志。此位指示是否发生过错误指令复位。

读出：

0 = 未发生过

1 = 发生过

写入：

1 = 清除该标志

第 5 位　　IARF：地址陷阱复位标志。此位指示是否发生过地址陷阱复位。

读出：

0 ＝ 未发生过

1 ＝ 发生过

写入：

1 ＝ 清除该标志

第 4 位　　保留

第 3 位　　LVRF：低电压复位标志。此位指示是否发生过低电压复位。

读出：

0 ＝ 未发生过

1 ＝ 发生过

写入：

1 ＝ 清除该标志

第 2 位　　WDRF：看门狗复位标志。此位指示是否发生过看门狗复位。

读出：

0 ＝ 未发生过

1 ＝ 发生过

写入：

1 ＝ 清除该标志

第 1 位　　PORF：上电复位标志。此位指示是否发生过上电复位。

读出：

0 ＝ 未发生过

1 ＝ 发生过

写入：

1 ＝ 清除该标志

第 0 位　　EXTRF：外部复位标志。此位指示是否发生过外部复位。

读出：

0 ＝ 未发生过

1 ＝ 发生过

写入：

1 ＝ 清除该标志

4.5 设计参考

4.5.1 外部复位

通用的复位电路如图 4-7 所示：

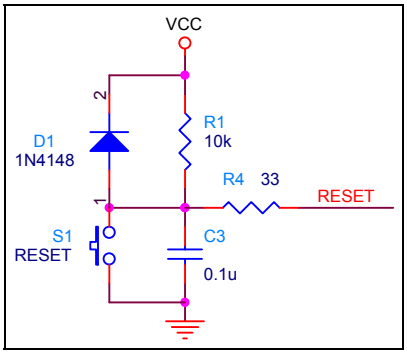


图 4-7 复位电路举例

为了设计出适当的复位电路，用户必须考虑电源和外部复位管脚上的干扰，因此推荐使用如下(图 4-8)所示的电路。

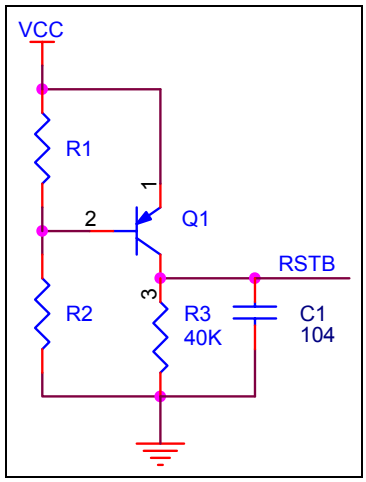


图 4-8 外部复位电路

注：当 VCC 电源降低到一定程度时，三极管 Q1 反相：VCC×R1/(R1+R2) = 0.7 V。如果 VDD = 2.5V，则可选用 R1= 140K ，R2 = 360K.。对于耗电敏感的芯片需要注意，这个电路本身将会消耗一部分电流，因为 R1 和 R2 组成了一个直流路径，电流消耗为：VDD/(R1+R2) = 3.3V(140K+360K) = 6.6 uA 。

4.5.2 复位状态

复位状态标识了复位的种类，用户可以通过这个寄存器查看复位的状态。若适当编程，系统就可以恢复复位之前的状态。用户可以在编程之前加入一段校验过程，如下：

```
【例 4-3】
if(P_Reset_Status->B.IARF & CB_CLEAR_IARF)    /* 查询 IARF 复位标志 */
{
P_Reset_Status->W = 0x5A5A;                   /* 清除复位状态标志 */
P_Reset_Status->B.IARF = CB_CLEAR_IARF;       /* 清除 IARF 复位标志*/
}
```

例 4-1 非法地址复位源的设计参考

4.6 启动引导序列

上电时系统会读出位于 Flash 信息区的第一个选项字(地址 = 0x8000)。上电时，上电复位功能有效，直到上电计数器计数到 16384 个辅助时钟周期时，复位请求被解除。详情请查看第 6 章的 P_System_Option 设置。

第5章 CPU

5.1 简介

SPMC75X 系例微控制器使用凌阳创新科技公司的μ'nSP™内核。μ'nSP™是一种高效的 16 位 RISC 内核，具有 4M Words 的寻址空间，支持多种寻址方式；支持乘法、乘累加、32/16 位除法、FIR 等运算；支持两种中断模式；内核拥有 2 个通用寄存器组，用户可以通过指令选择当前的工作寄存器组。可以实现如三相的 SPWM、空间向量变换（SVPWM 合成）等算法。其结构如图 5-1 所示。

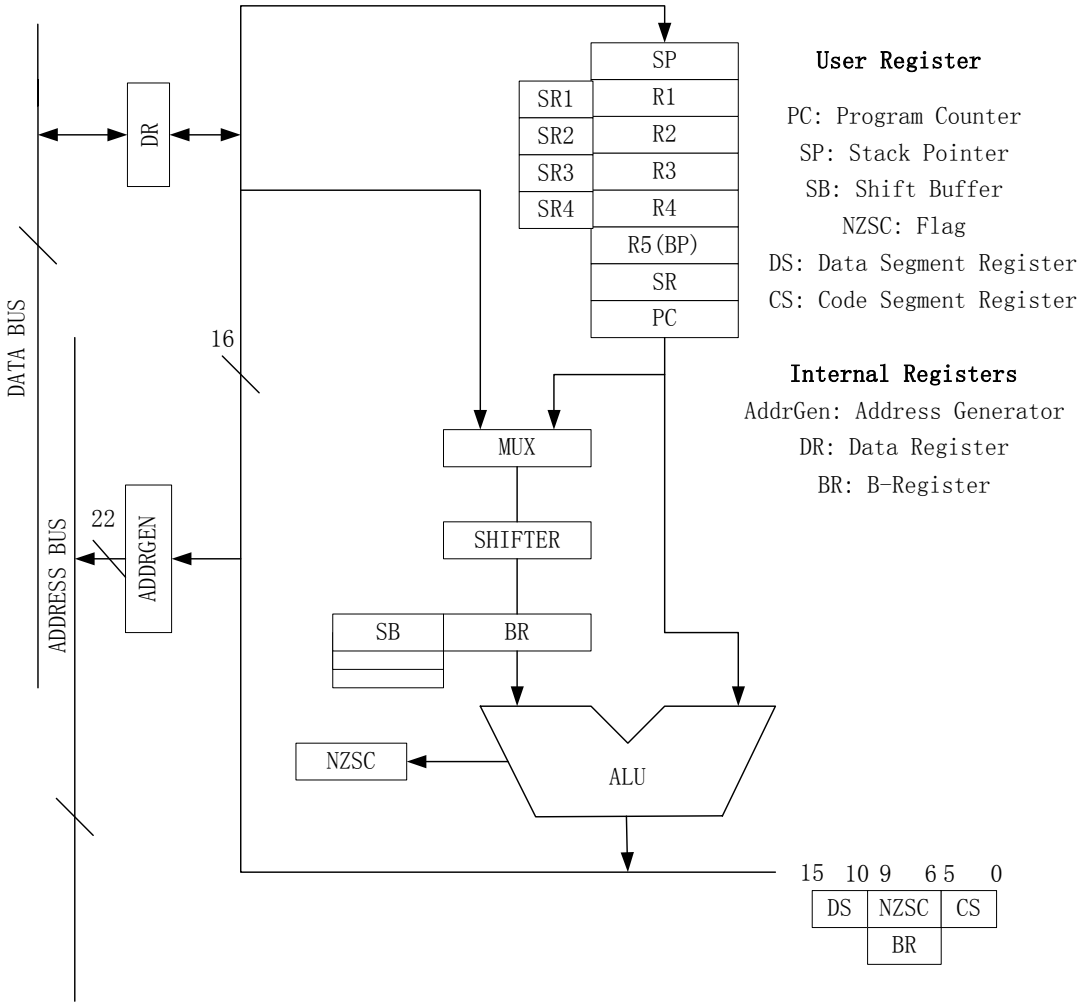


图 5-1 μ'nSP™1.2 内核结构

5.2 CPU 内部寄存器

μ'nSP™内核有 13 个 16 位内部寄存器：程序指针 PC、堆栈指针 SP、状态标志寄存器 SR、CPU 标志寄存器 FR、基址寻址寄存器 BP（R5）、通用寄存器 R1-R4 和辅助通用寄存器 SR1-SR4。CPU 寄存器详述见表 5-1:

表 5-1　CPU 寄存器

寄存器	长度	描述
PC	16Bit	程序计数器 PC(Program Counter)：指向 CPU 即将执行的下一条指令的地址
SP	16Bit	堆栈指针，SP 中的内容指示当前堆栈的栈顶。
SR	16Bit	状态标志寄存器 SR：包括 N、S、Z、C、DS（数据段）和 CS（程序段）
FR	16Bit	CPU 状态标志寄存器。包括 AQ、BNK、FRA、FIR 等状态标志
BP	16Bit	基址指针寄存器 BP，变址寻址的基地址
R1-R4	16Bit	通用寄存器，通常可分别用于数据运算或传送的源及目标寄存器。
SR1-SR4	16Bit	辅助通用寄存器 SR1-SR4：与 R1-R4 的功能一样，SECBNK ON 后有效

5.2.1　程序计数器 PC

程序计数器 PC(Program Counter)：它的作用与其它微控制器(MCU)中的 PC 作用相同，是作为程序的地址指针的专用寄存器。当 CPU 完成一条指令后，PC 会累加这条指令所占的字数，以指向下一条指令地址。在μ'nSP™中，PC 通常与 SR 寄存器中的 CS 字段共同组成 22 位的代码地址。

5.2.2　堆栈指针 SP

SP 中的内容指示当前堆栈段内的栈顶地址。CPU 执行压栈/出栈指令(push/pop)、子程序调用/返回指令(call/retf)以及进入中断服务子程序 ISR 或从 ISR 返回指令(reti)时自动减少/增加，以示堆栈指针的移动。堆栈的最大容量范围限制在 2K 字的 RAM 内，即地址为 0x000000-0x0007FF 的存储器中。

5.2.3　状态标志寄存器 SR

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
DS						N	Z

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

S	C	CS

数据段 DS： 与 16 位的寄存器组合在一起形成 22 位地址码，用来寻址 4M 字容量的存储器。DS 可通过对 SR 的赋值来改变

程序段 CS： 与 16 位的寄存器组合在一起形成 22 位地址码，用来寻址 4M 字容量的存储器。CS 值只能通过调用指令 Call、长跳转指令 GOTO 改变

进位标志 C： 当 C=0 时，表示运算过程中无进位或借位产生；当 C=1 时，表示运算过程中有进位或借位产生。在无符号数运算中，16 位数可以表示的数值范围是 0x0000-0xFFFF, 即 0-65535。如果运算结果大于 0xFFFF，则标志位 C 被置 1

符号位 S： 对于有符号数运算，当 S=0 时，表示实际运算结果非负(包括正数和零)；当 S=1 时，表示实际运算结果为负数。对于有符号数运算，16 位数所表示的数值范围为 0x8000-0x7FFF， 即-32768-32767。若运算结果为负数，则标志位 S 置 1

零标志 Z： 当 Z=0 时，表示运算结果不为 0；当 Z=1 时，表示运算结果为 0

负标志 N： 当 N=0 时，表示运算结果的最高位(B15)为 0；当 N=1 时，表示运算结果的最高位(B15)为 1

根据标志位 N 和 S，可判断运算结果是否溢出。若标志位 N 和 S 不同，即当 S=0，N=1 或 S=1，N=0 时，则说明运算结果溢出。

5.2.4 CPU 状态标志寄存器 FR

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留	AQ	BNK	FRA	FIR	SFTBUF		

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SFTBUF	F	I	INE	IRQ PRI			

AQ ： DIVS/DIVQ 除法指令执行的 AQ 标志，默认为零

BNK： 第二寄存器组模式标志，默认为零(SECBNK ON：1， SECBNK OFF：0)

FRA： FRACTION MODE，默认为零(FRACTION MODE ON：1，FRACTION MODE OFF：0)

FIR： FIR_MOVE MODE，默认为零(FIR_MOVE ON：0，FIR_MOVE OFF：1)

SFTBUF： 移位缓冲器或者 FIR 的保护位，默认为 “0000”B

F： FIQ 中断标志，默认为零

I： IRQ 中断标志，默认为零

INE： IRQ 嵌套模式标志，默认为零(IRQNEST ON：1，IRQNEST OFF：0)

IRQ PRI： IRQ 优先级寄存器，复位后默认为“1000B”,如果发生任何中断，IRQ PRI 寄存器将被设为 IRQ 的优先级，只有具有更高优先级的 IRQ 才可将其中断，用户可以通过设定优先级寄存器来定制 IRQ 嵌套。

优先级顺序： IRQ0>IRQ1>IRQ2>IRQ3>IRQ4>IRQ5>IRQ6>IRQ7

IRQ 使能算法： 0 - PRI -1

注：如果FIQ使能，FIQ依然具有高于任何IRQ的优先级。

例如：

1. 如果PRI为1000, 所有IRQ 0-7使能
2. 如果PRI为0000, 所有IRQ 0-7均被禁止

5.2.5 基址指针寄存器 BP

基址指针寄存器 BP(Base Pointer)：BP 可用来存放当前 Bank 数据区的基地址偏移量，它可和一个 6 位的立即数（CS 或是 DS）组成 22 位数据地址，从存储器中存取数据。除此之外，BP 也可作为一般用户寄存器 R5 使用，完成与其它一般寄存器相同的工作。

5.2.6 通用寄存器 R1-R4

一般寄存器 R1-R4：通常可分别用于数据运算或传送的源及目标寄存器。其中 R3、R4 还可组成一个 32 位的结果寄存器 MR，用于存放乘法运算和内积运算之结果(R4 存放高位字)。

5.2.7 辅助通用寄存器 SR1-SR4

辅助通用寄存器 SR1-SR4：SECBNK ON 指令后，所有对 R1-R4 的操作全部映射到 SR1-SR4，SECBNK OFF 后返回正常状态。

5.3 CPU 寻址方式

μ'nSP™内核支持以下 6 种寻址方式，所有寻址方式均适合整个 CPU 寻址空间：

- 立即数寻址
- 直接寻址
- 寄存器寻址
- 寄存器间接寻址
- 寄存器间接增量寻址
- 基址变址引址
- 多重间接寻址

5.3.1 立即数寻址

这种寻址方式是操作数以立即数的形式出现，立即数有两种：6 位立即数(用 IM6 表示，范围 0x00-0x3F)和 16 位立即数(用 IM16 表示，范围 0x0000-0xFFFF)。

例如：

R2 = 0x32： 是把 6 位立即数 0x32 赋给寄存器 R2。这条指令是单字指令，操作数在操作码的低 6 位。如图 5-2 的 a 图所示

R2 = 0x1032：是把 16 位立即数 0x1032 赋给寄存器 R2。这条指令是双字指令，操作数存在紧随操作码的单元中。如图 5-2 的 b 图所示

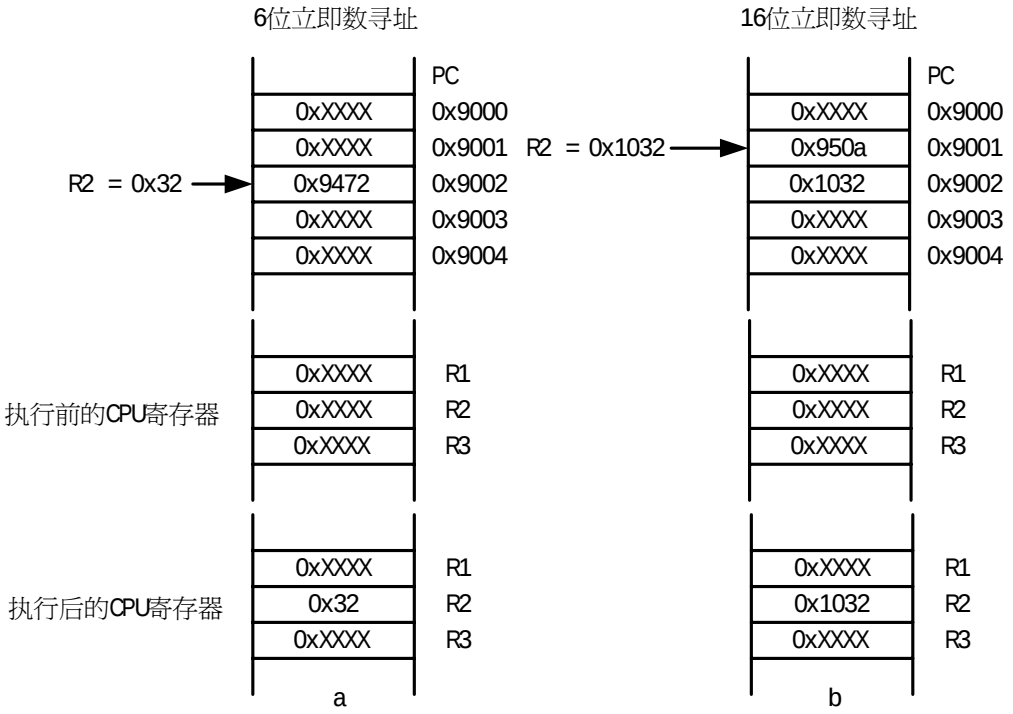


图 5-2 立即数寻址操作示意

5.3.2 直接寻址

直接寻址的操作地址由指令直接给出。根据地址长度，直接寻址操作的地址分两种：6 位地址(用 IM6 表示，范围 0x00-0x3F)和 16 位地址(用 IM16 表示，范围 0x0000-0xFFFF)。

例如：

[0x0020] = R2： 将 R2 中的内容存到 0x0020 单元中。这条指令是单字指令，6 位立即地址存在操作码的低 6 位。如图 5-3 的 a 图所示

[0x0200] = R2： 将 R2 中的内容存到 0x0200 单元中。这条指令是双字指令，16 位立即地址存在紧随操作码的单元中。如图 5-3 的 b 图所示

R2 = [0x0020]： 将 0x0020 单元中的内容存到 R2 中。这条指令是单字指令，6 位立即地址存在操作码的低 6 位。如图 5-4 的 a 图所示

R2 = [0x0200]： 将 0x0200 单元中的内容存到 R2 中。这条指令是双字指令，16 位

立即地址存在紧随操作码的单元中。如图 5-4 的 b 图所示

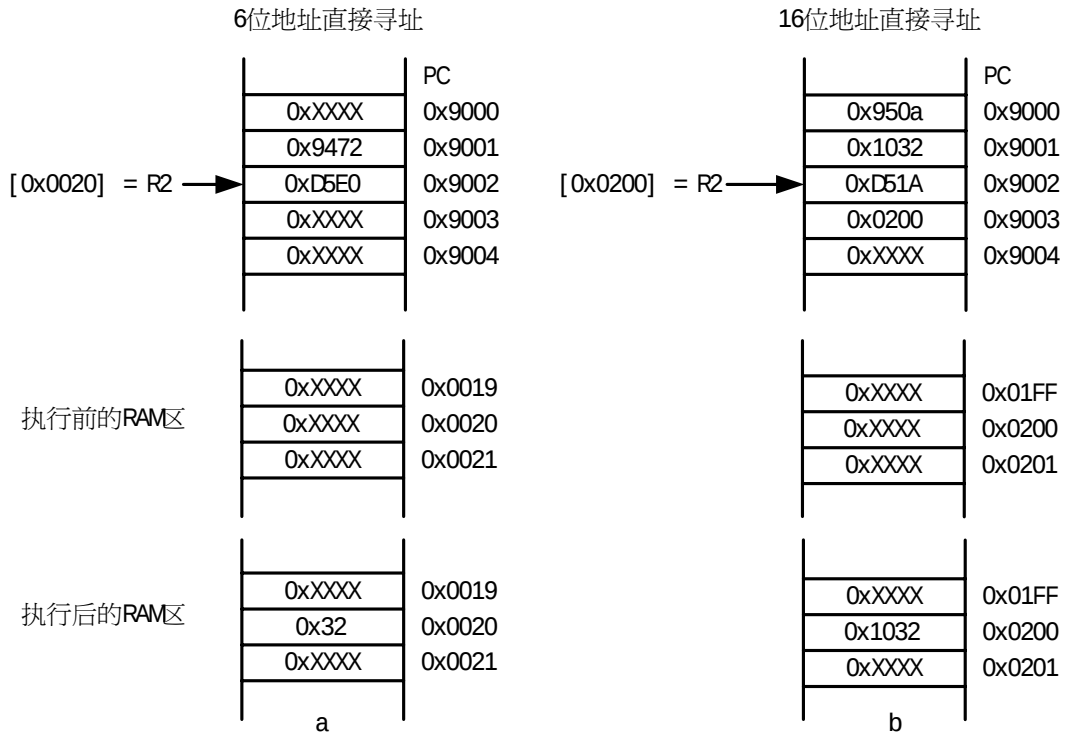


图 5-3 直接寻址操作示意 1

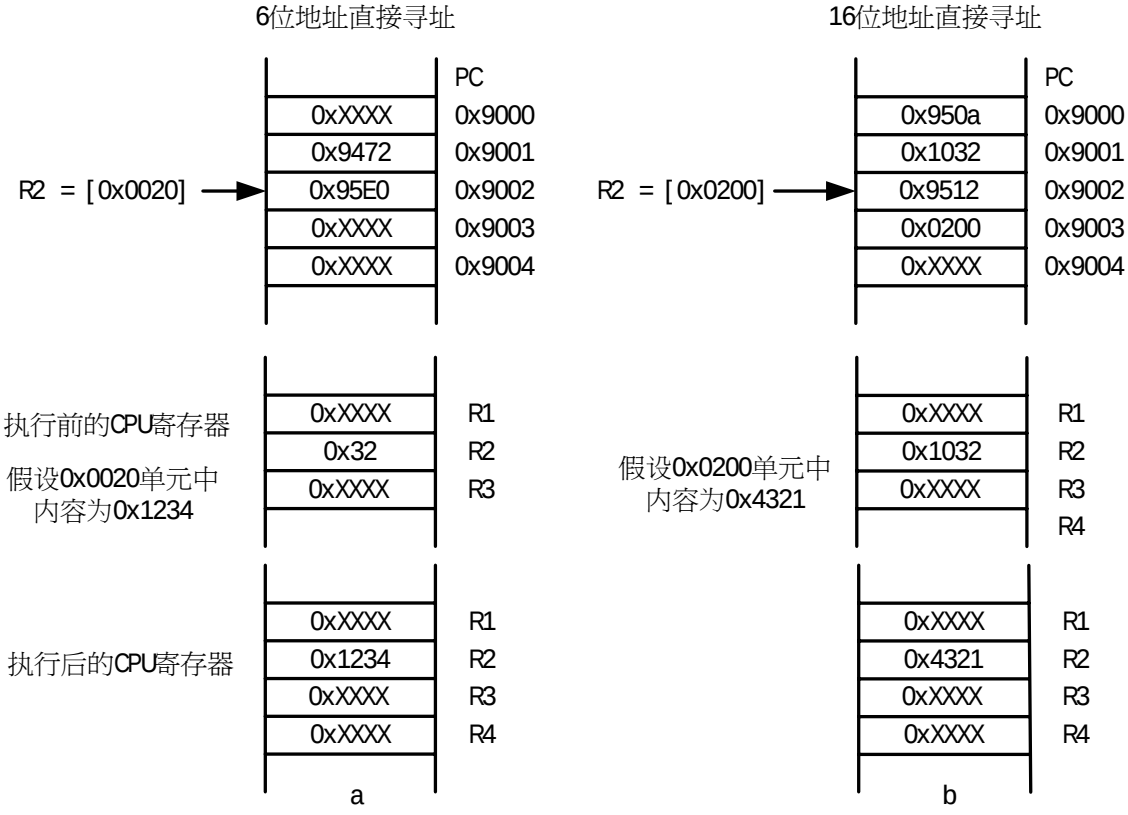


图 5-4 直接寻址操作示意 2

5.3.3 寄存器寻址

将源寄存器的内容送到目标寄存器中，源寄存器内容不变。

例如：

R1 = R2： 是把寄存器 R2 中的内容赋给寄存器 R1。这条指令是单字指令，目标寄存器和源寄存器的 ID 均隐含在操作码中。如图 5-5 所示

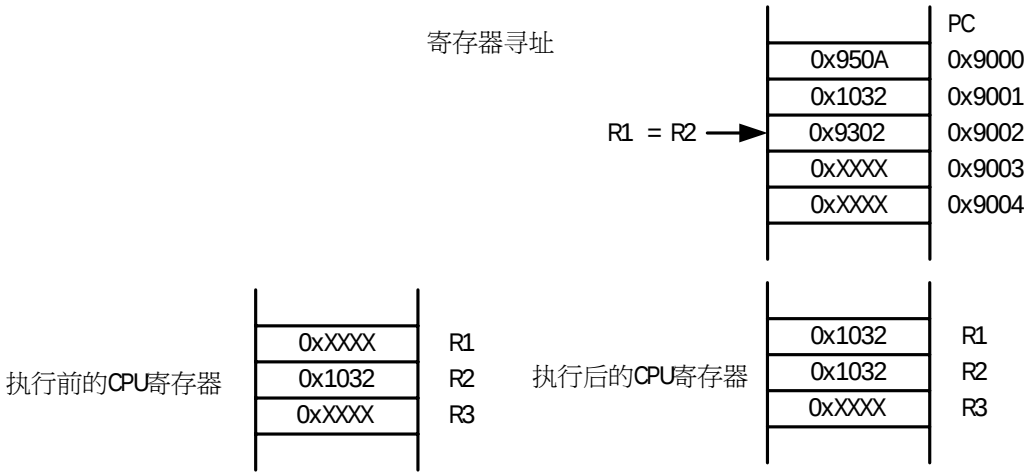


图 5-5 寄存器寻址操作示意

5.3.4 寄存器间接寻址

寄存器间接寻址的操作数地址由寄存器给出，即将寄存器作为指针指向操作地址。

例如：

R1 = [R2]，是把由 R2 指向的内存单元的数据送寄存器 R1，如图 5-6 所示

[R2] = R1，将 R1 中的内容存到 R2 所指向的内存单元中，如图 5-7 所示

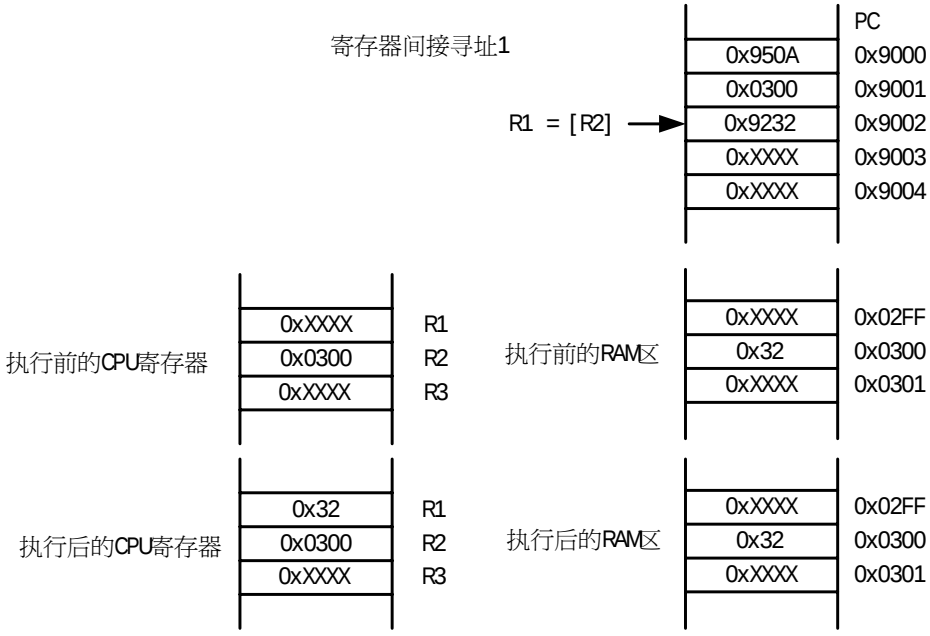


图 5-6 寄存器间接寻址操作示意 1

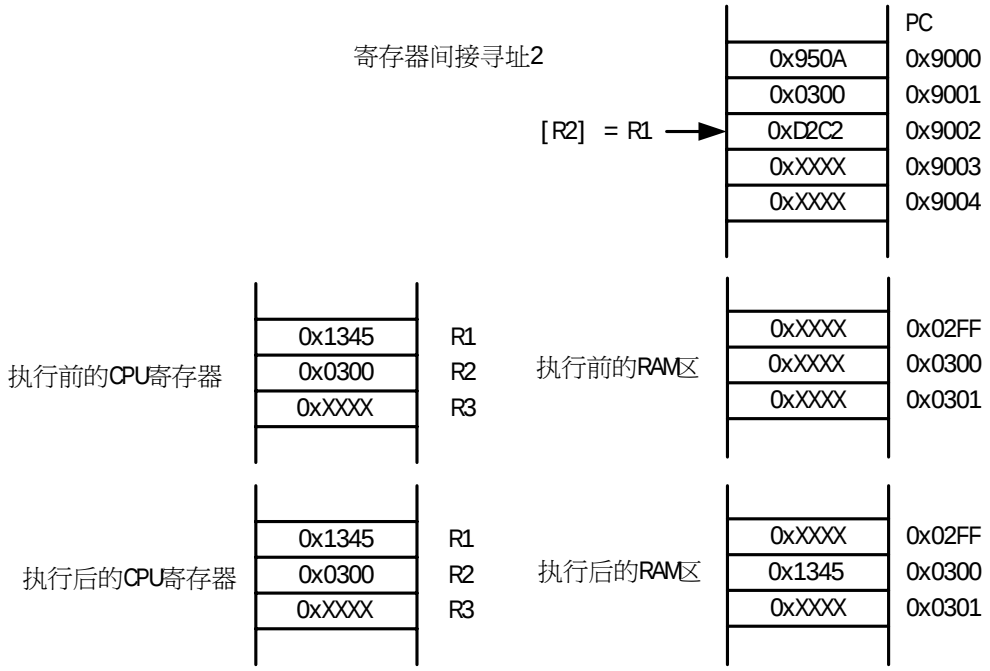


图 5-7 寄存器间接寻址操作示意 2

5.3.5 寄存器间接增量寻址

这种寻址方式比寄存器间接寻址多了指针（寄存器）自动调整功能，其它操作完全相同。

例如：

R1 = [R2 --]，是把由 R2 指向的内存单元的数据送寄存器 R1，完成后将 R2 自动减 1

[R2 --] = R1，将 R1 中的内容存到 R2 所指向的内存单元中，完成后将 R2 自动减 1

R1 = [R2 ++]，是把由 R2 指向的内存单元的数据送寄存器 R1，完成后将 R2 自动加 1

[R2 ++] = R1，将 R1 中的内容存到 R2 所指向的内存单元中，完成后将 R2 自动减 1

[++ R2] = R1，先将 R2 自动加 1，而后将 R1 中的内容存到 R2 所指向的内存单元中

R1 = [++ R2]，先将 R2 自动加 1，而后把由 R2 指向的内存单元的数据送寄存器 R1

5.3.6 基址变址寻址

这种寻址方式下，操作数的地址由基址寄存器 BP 和 6 位（0x00-0x3F）偏移量相加后给出。

例如：

R1 = [BP+0x32]，将 BP+0x32 所指向的内存单元中的内容送到 R1 中。如图 5-8 所示

[BP+0x34] = R1，将 R1 中的内容送到 BP+0x32 所指向的内存单元中。如图 5-9 所示

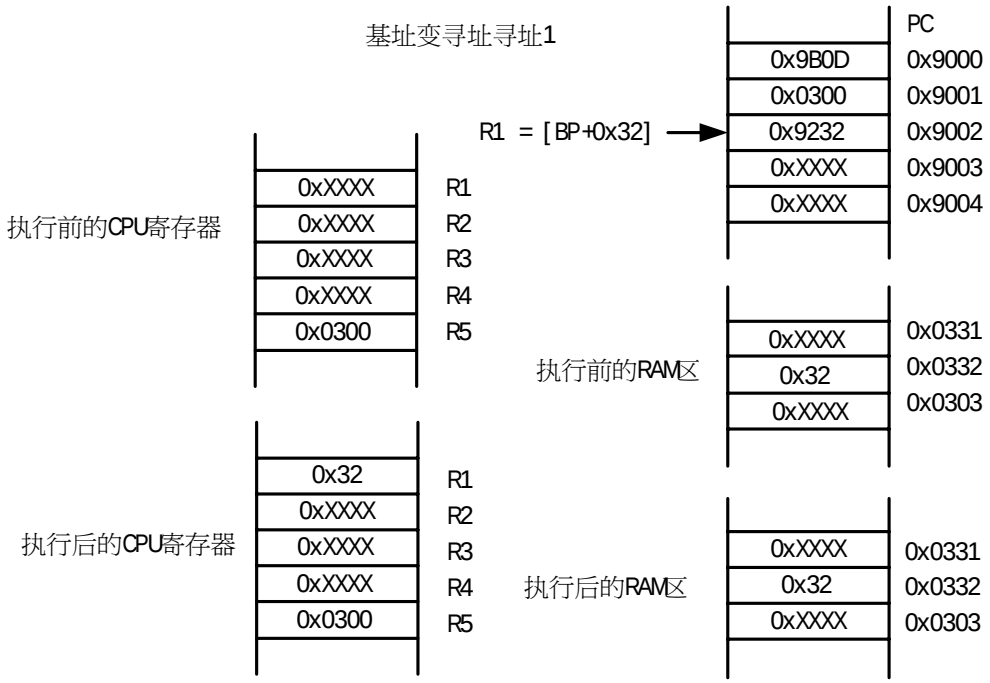


图 5-8 基址变址寻址操作示意 1

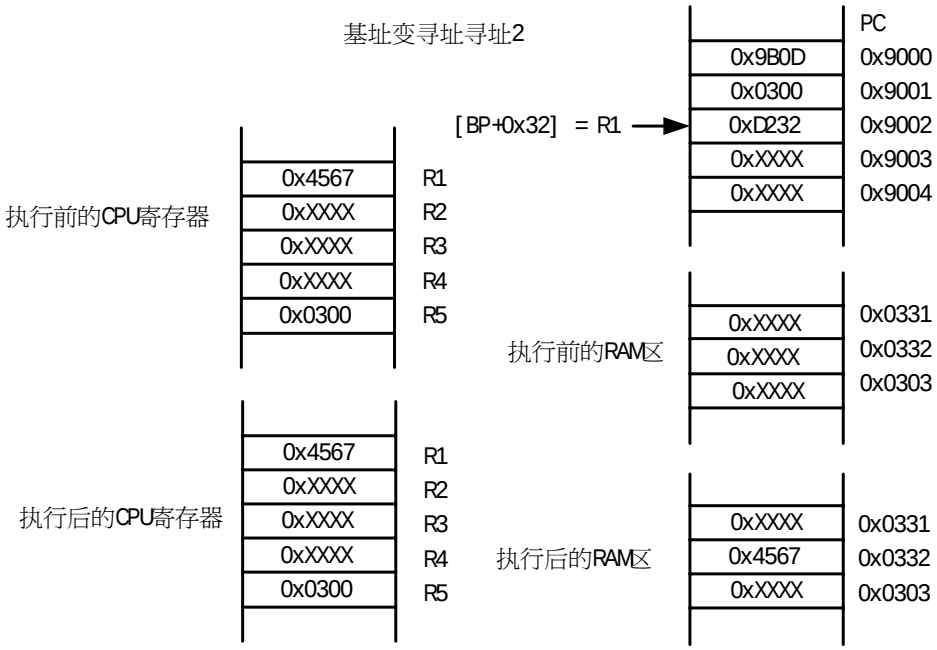


图 5-9 基址变址寻址操作示意 2

5.3.7 多重间接寻址

这种寻址方式下，用户可以将多个寄存器的值一次压入堆栈，或从堆栈中弹出。

例如：

PUSH R1,R4 TO [SP]，是将 R1、R2、R3 和 R4 寄存器的值压入堆栈，如图 5-10 所示

POP R1,R4 TO [SP]， 堆栈中的值弹出到 R1、R2、R3 和 R4 寄存器中，如图 5-11 所示

注：多重间接寻址并不只针对堆栈操作，R1-R5 都可以作为其中的指针（如：PUSH R1,R4 TO [R5]和 POP R1,R4 TO [R5]）。

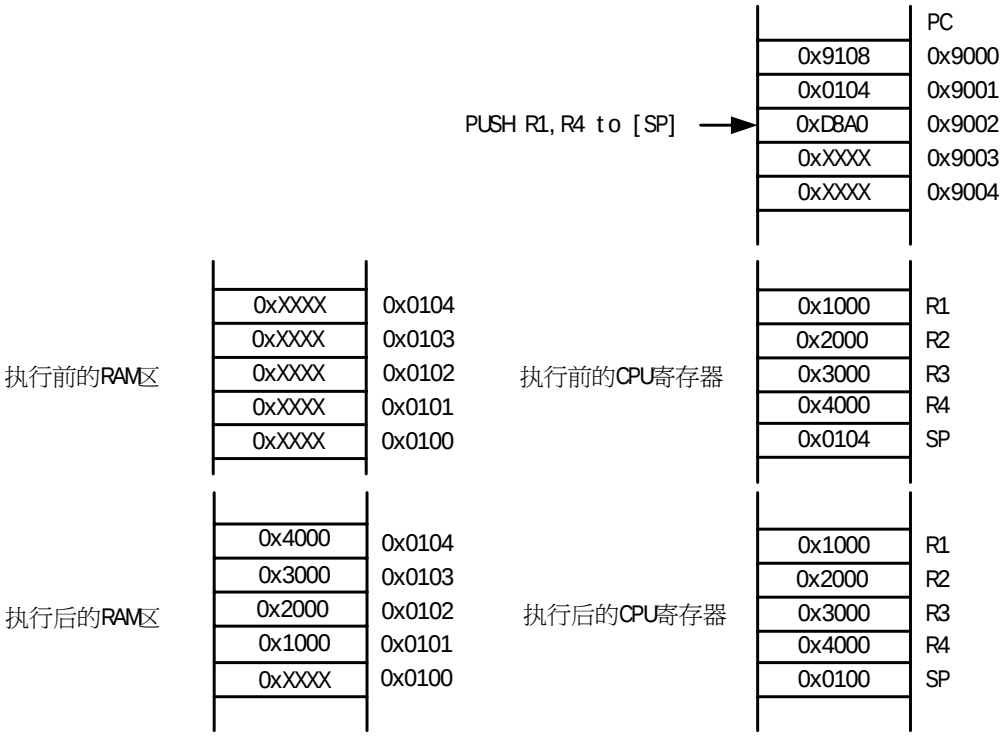


图 5-10 多重间接寻址操作示意 1

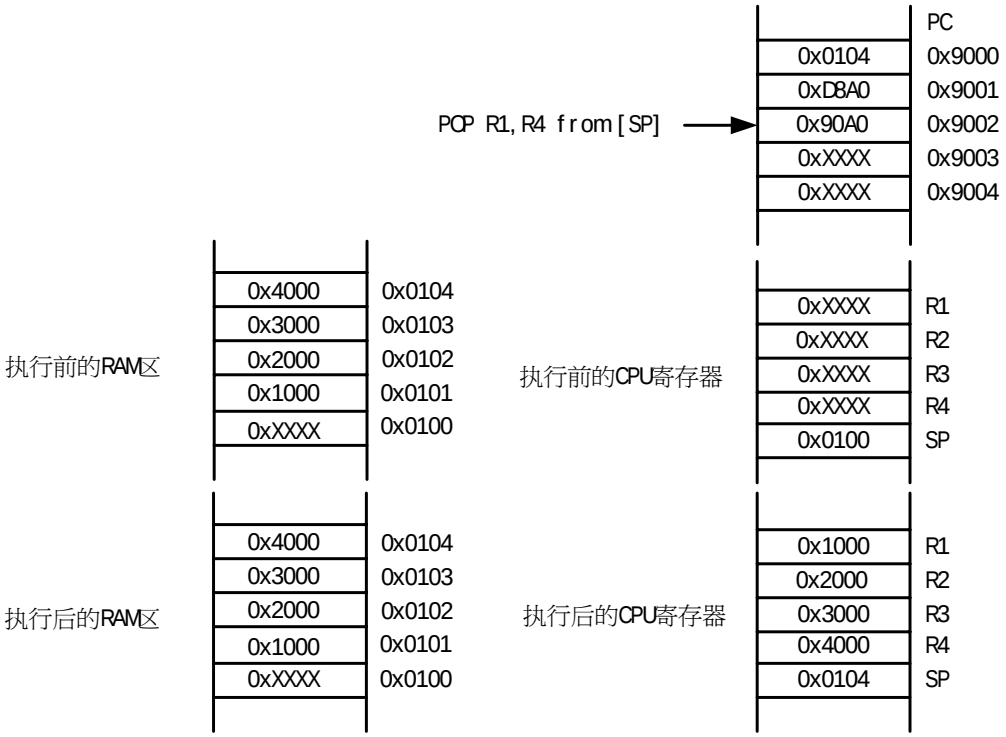


图 5-11 多重间接寻址操作示意 2

第6章 存储结构

6.1 简介

SPMC75X 系列微控制器的所有空间统一编址。其内部存储空间分为 Flash、SRAM 和外设控制寄存器三部分。

- 32K 字的片内 Flash，分为信息区和通用区
- 2K 字的片内 SRAM，用作堆栈空间和数据存贮
- 4K 字的外设控制寄存器空间

SPMC75X 系列微控制器的存贮空间分配如图 6-1 所示：

- SRAM：
  占用 0x0000-0x07FF
- 外设控制寄存器：
  占用 0x7000-0x7FFF
- Flash 信息区：
  占用 0x8000-0x803F
- Flash 通用区：
  占用 0x8040-0xFFFF

图 6-1 存贮空间分配

6.2 Flash

SPMC75X 系列微控制器 Flash 分为两区：信息区和通用区，在同一时间只能访问其中的一区。信息区包含 64 个字，寻址空间为 0x8000～0x803F。地址 0x8000 为系统选项寄存器 P_System_Option。其他地址空间可由用户自定义重要信息比如：版本控制、日期、版权、项目名称等等。信息区的内容只有在仿真或烧录的状态下才能改变。

32k 字的内嵌 Flash(embedded Flash)被划分为 16 个页，每页 2K 字，每页可分为 8 块，这样 32K 的 Flash 就可以分成 128 个块。只有位于 00F000～00F7FF 区域的页在自由运行模式下可以设置为只读或可读可写，其他页均为只读。

6.3 SRAM

SPMC75X 系列微控制器中的 SRAM 用于堆栈和变量和数据的存储。堆栈用于存放调用函数的返回地址和入栈数据。 堆栈的增长方向为自下向顶，称为 FILO(先进后出)结构，堆栈的地址由堆栈指针(SP)来指示。在 SRAM 中的数据存储是由用户设定的，可以直接访问，间接访问或用指针访问。需要注意的是堆栈区与数据存储区千万不能交迭，否则会发生系统崩溃。SPMC75X 系列微控制器对 SRAM 最大可寻址空间为从 0x0000 到 0x07FF 共 2K 字，堆栈指针 SP 最大允许指向 0x07FF。

6.4 外设控制寄存器

SPMC75X 系列微控制器的 CPU 通过外设控制寄存器完成对相应外设模块的控制。SPMC75X 系列微控制器的外设控制寄存器分布在 0x7000-0x7FFF 4K 字的存贮空间中，SPMC75X 系列微控制器所有外设控制寄存器地址和名称如表 6-1 所示，其它地址为保留地址。控制寄存器的具体功能描述请参考各功能模块的介绍。

表 6-1 描述了全部的 I/O 口寄存器地址和名称

IO 端口	地址	IO 端口	地址	IO 端口	地址	IO 端口	地址
保留	$7000	保留	$7010	P_IOA_Data	$7060	P_IOC_Data	$7070
	$7001		$7011	P_IOA_Buffer	$7061	P_IOC_Buffer	$7071
	$7002		$7012	P_IOA_Dir	$7062	P_IOC_Dir	$7072
	$7003		$7013	P_IOA_Attrib	$7063	P_IOC_Attrib	$7073
	$7004		$7014	P_IOA_Latch	$7064	保留	$7074
	$7005		$7015	保留	$7065		$7075
P_Reset_Status	$7006		$7016		$7066		$7076
P_Clk_Ctrl	$7007		$7017		$7067		$7077
保留	$7008		$7018	P_IOB_Data	$7068	P_IOD_Data	$7078
	$7009		$7019	P_IOB_Buffer	$7069	P_IOD_Buffer	$7079
P_WatchDog_Ctrl	$700A		$701A	P_IOB_Dir	$706A	P_IOD_Dir	$707A
P_WatchDog_Clr	$700B		$701B	P_IOB_Attrib	$706B	P_IOD_Attrib	$707B
P_Wait_Enter	$700C		$701C	保留	$706C	保留	$707C
保留	$700D		$701D		$706D		$707D
P_Stdby_Enter	$700E		$701E		$706E		$707E
保留	$700F		$701F		$706F		$707F

IO 端口	地址	IO 端口	地址	IO 端口	地址	IO 端口	地址
P_IOA_SPE	$7080	保留	$7090	P_INT_Status	$70A0	保留	$70B0
P_IOB_SPE	$7081		$7091	保留	$70A1		$70B1
P_IOC_SPE	$7082		$7092		$70A2		$70B2
保留	$7083		$7093		$70A3		$70B3
P_IOA_KCER	$7084		$7094	P_INT_Priotity	$70A4		$70B4
保留	$7085		$7095	保留	$70A5		$70B5

IO 端口	地址	IO 端口	地址	IO 端口	地址	IO 端口	地址
	$7086		$7096		$70A6		$70B6
	$7087		$7097		$70A7		$70B7
	$7088		$7098	P_MisINT_Ctrl	$70A8	P_TMB_Start	$70B8
	$7089		$7099		$70A9	P_BZO_Ctrl	$70B9
	$708A		$709A		$70AA		$70BA
	$708B		$709B		$70AB		$70BB
	$708C		$709C	保留	$70AC	保留	$70BC
	$708D		$709D		$70AD		$70BD
	$708E		$709E		$70AE		$70BE
	$708F		$709F		$70AF		$70BF

IO 端口	地址	IO 端口	地址	IO 端口	地址	IO 端口	地址
P_UART_Data	$7100	P_SPI_Ctrl	$7140	P_ADC_Setup	$7160		$7170
P_UART_RxStatus	$7101	P_SPI_TxStatus	$7141	P_ADC_Ctrl	$7161		$7171
P_UART_Ctrl	$7102	P_SPI_TxBuf	$7142	P_ADCData	$7162		$7172
P_UART_BaudRate	$7103	P_SPI_RxStatus	$7143		$7163		$7173
P_UART_Status	$7104	P_SPI_RxBuf	$7144	保留	$7164		$7174
	$7105		$7145		$7165		$7175
	$7106		$7146	P_ADC_Channel	$7166		$7176
	$7107		$7147		$7167	保留	$7177
	$7108		$7148		$7168		$7178
	$7109		$7149		$7169		$7179
保留	$710A	保留	$714A		$716A		$717A
	$710B		$714B	保留	$716B		$717B
	$710C		$714C		$716C		$717C
	$710D		$714D		$716D		$717D
	$710E		$714E		$716E		$717E
	$710F		$714F		$716F		$717F

IO 端口	地址	IO 端口	地址	IO 端口	地址	IO 端口	地址
P_TMR0_Ctrl	$7400	P_TMR0_IOCtrl	$7410	P_TMR0_INT	$7420	P_TMR0_TCNT	$7430
P_TMR1_Ctrl	$7401	P_TMR1_IOCtrl	$7411	P_TMR1_INT	$7421	P_TMR1_TCNT	$7431
P_TMR2_Ctrl	$7402	P_TMR2_IOCtrl	$7412	P_TMR2_INT	$7422	P_TMR2_TCNT	$7432
P_TMR3_Ctrl	$7403	P_TMR3_IOCtrl	$7413	P_TMR3_INT	$7423	P_TMR3_TCNT	$7433
P_TMR4_Ctrl	$7404	P_TMR4_IOCtrl	$7414	P_TMR4_INT	$7424	P_TMR4_TCNT	$7434
P_TMR_Start	$7405	保留	$7415	P_TMR0_Status	$7425	P_TMR0_TPR	$7435
P_TMR_Output	$7406		$7416	P_TMR1_Status	$7426	P_TMR1_TPR	$7436
P_TMR3_OutputCtrl	$7407		$7417	P_TMR2_Status	$7427	P_TMR2_TPR	$7437
P_TMR4_OutputCtrl	$7408		$7418	P_TMR3_Status	$7428	P_TMR3_TPR	$7438
P_TPMW_Write	$7409		$7419	P_TMR4_Status	$7429	P_TMR4_TPR	$7439
P_TMR_LDOK	$740A		$741A	保留	$742A	保留	$743A

IO 端口	地址	IO 端口	地址	IO 端口	地址	IO 端口	地址
保留	$740B		$741B		$742B		$743B
	$740C		$741C		$742C		$743C
	$740D		$741D		$742D		$743D
	$740E		$741E		$742E		$743E
	$740F		$741F		$742F		$743F

IO 端口	地址	IO 端口	地址	IO 端口	地址	IO 端口	地址
P_TMR0_TGRA	$7440	P_TMR0_TBRA	$7450	P_TMR3_DeadTime	$7460	P_CMT_Start	$7500
P_TMR0_TGRB	$7441	P_TMR0_TBRB	$7451	P_TMR4_DeadTime	$7461	P_CMT_Ctrl	$7501
P_TMR0_TGRC	$7442	P_TMR0_TBRC	$7452	P_POS0_DectCtrl	$7462	保留	$7502
P_TMR1_TGRA	$7443	P_TMR1_TBRA	$7453	P_POS1_DectCtrl	$7463		$7503
P_TMR1_TGRB	$7444	P_TMR1_TBRB	$7454	P_POS0_DectData	$7464		$7504
P_TMR1_TGRC	$7445	P_TMR1_TBRC	$7455	P_POS1_DectData	$7465		$7505
P_TMR2_TGRA	$7446	P_TMR2_TBRA	$7456	P_Fault1_Ctrl	$7466		$7506
P_TMR2_TGRB	$7447	P_TMR2_TBRB	$7457	P_Fault2_Ctrl	$7467		$7507
P_TMR3_TGRA	$7448	P_TMR3_TBRA	$7458	P_OL1_Ctrl	$7468	P_CMT0_TCNT	$7508
P_TMR3_TGRB	$7449	P_TMR3_TBRB	$7459	P_OL2_Ctrl	$7469	P_CMT1_TCNT	$7509
P_TMR3_TGRC	$744A	P_TMR3_TBRC	$745A	P_Fault1_Release	$746A	保留	$750A
P_TMR3_TGRD	$744B	保留	$745B	P_Fault2_Release	$746B		$750B
P_TMR4_TGRA	$744C	P_TMR4_TBRA	$745C	保留	$746C		$750C
P_TMR4_TGRB	$744D	P_TMR4_TBRB	$745D		$746D		$750D
P_TMR4_TGRC	$744E	P_TMR4_TBRC	$745E		$746E		$750E
P_TMR4_TGRD	$744F	保留	$745F		$746F		$750F

IO 端口	地址	IO 端口	地址
P_CMT0_TPR	$7510	保留	$7550
P_CMT1_TPR	$7511		$7551
保留	$7512		$7552
	$7513		$7553
	$7514		$7554
	$7515	P_Flash_CMD	$7555
	$7516	保留	$7556
	$7517		$7557
	$7518		$7558
	$7519		$7559
	$751A		$755A
	$751B		$755B
	$751C		$755C
	$751D		$755D
	$751E		$755E
	$751F		$755F

6.5 复位和中断入口地址

SPMC75X 系列微控制器一共有 10 个中断入口地址，分布在 0xFFF5-0xFFFF 这几个地址。其名称及地址见表 6-2。

表 6-2 中断入口地址表

中断源	中断入口地址
BREAK	0xFFF5
FIQ	0xFFF6
Reset	0xFFF7
IRQ0	0xFFF8
IRQ1	0xFFF9
IRQ2	0xFFFA
IRQ3	0xFFFB
IRQ4	0xFFFC
IRQ5	0xFFFD
IRQ6	0xFFFE
IRQ7	0xFFFF

第7章 Flash 的存储和控制

7.1 简介

SPMC75X 系列微控制器的 Flash 分为两区：信息区和通用区，在同一时间只能访问其中的一区。信息区包含 64 个字，寻址空间为 0x8000～0x803F。地址 0x8000 为系统选项寄存器 P_System_Option，其余空间可由用户自定义一些重要信息比如：版本控制，日期，版权名称，项目名称等等。 信息区的结构见图 7-1，信息区的内容只有在仿真或烧录的状态下才能改变。32k 内嵌的 Flash 被划分为 16 页，每页 2K 字，每页再分为 8 个块，共可分成 128 个块。只有位于 00F000～00F7FF 区域的页在自由运行模式下可以设置为只读或可读可写，其他页均为只读。用户可以分别对每块进行擦写操作。Flash 的页和块的关系见图 7-2。

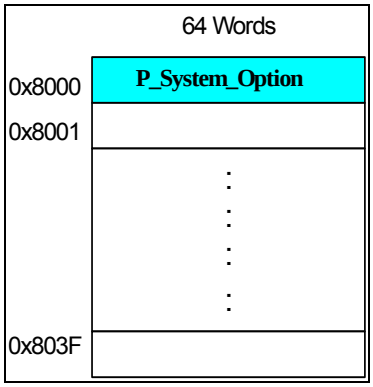


图 7-1 信息区的结构图

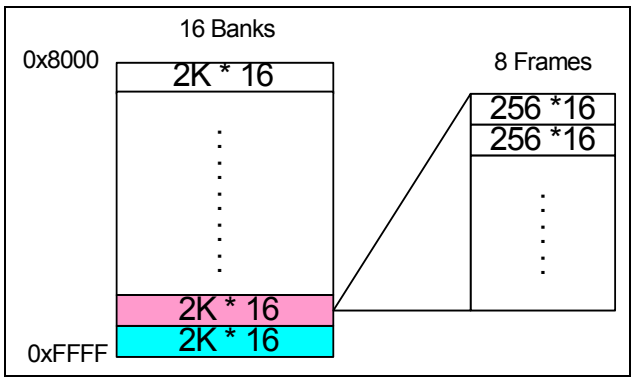


图 7-2 Flash 的页和块

7.2 控制寄存器

SPMC75X 系列微控制器的 Flash 模块共有 3 个控制寄存器，如表 7-1 所示。通过这 3 个控制寄存器可以完成 Flash 模块所有功能的控制。

表 7-1 Flash 模块控制寄存器表

地址	寄存器	名称
704Dh	P_Flash_RW	内嵌的 Flash 访问控制寄存器
7555h	P_Flash_CMD	内嵌的 Flash 控制寄存器
8000h	P_System_Option	系统选项寄存器

7.2.1 Flash 访问控制寄存器

P_Flash_RW 是 Flash 写功能使能寄存器。用控制寄存器将页设置为只读或可读可写模式。为避免误操作，对其写入必需连续两次写操作方能完成：首先向该寄存器写入$5a5a，然后在 16 个 CPU 时钟周期内再向该寄存器写入相应的设置字。

P_Flash_RW ($704D)：Flash 访问控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R/W	R	R	R	R	R	R
1	0	1	1	1	1	1	1
保留	BK14WENB	保留					

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
1	1	1	1	1	1	1	1
保留							

第 15 位　　保留

第 14 位　　BK14WENB：　第 14 位写使能标志位（0xF000h-0xF7FFh）

1= 只读

0= 读/写

第 13:0 位　　保留

7.2.2　Flash 控制命令寄存器

P_Flash_CMD 是 Flash 控制命令寄存器，只能进行写操作。所有对 Flash 的操作均通过向其写入相应的控制命令来完成。其命令总表见表 7-2 所示：

P_Flash_CMD ($7555)：内嵌的 Flash 控制命令寄存器

B15	B14	B13	B12	B11	B10	B9	B8
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
FlashCtrl							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
FlashCtrl							

表 7-2 指令功能和操作流程

	块擦除	单字写模式	连续多字写模式
第一步	P_Flash_CMD = 0xAAAA		
第二步	[P_Flash_CMD] = 0x5511	[P_Flash_CMD] = 0x5533	[P_Flash_CMD] = 0x5544
第三步	设置擦除地址	写数据	写数据
第四步	自动等待 20ms 后结束	自动等待 40us 后结束	自动等待 40us
			未写完则转向第二步
			[P_Flash_CMD]= 0xFFFF → 操作结束令

7.2.3 系统选项寄存器

该寄存器位于 Flash 的信息区，用户只能在 ICE 环境下或通过烧录来设置该寄存器。

P_System_Option ($8000)：系统选项寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	1	0	1	0	1	0	1
校验方式							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	1	0	1	0	1	1	1
校验方式			Security	保留	LVR	WDG	CLKS

第 15:5 位 **校验方式：**

在仿真或烧录时写入 01010101010

第 4 位 **Security**： 信息保护位

0 ＝信息保护，无法访问通用 Flash 区

1 ＝无信息保护，可读可写

第 3 位 保留

第 2 位 **LVR**：低电压复位功能使能

0 ＝ 禁止

1 ＝ 使能

第 1 位 **WDG**：看门狗使能

0 ＝ 禁止

1 ＝ 使能

第 0 位 **CLKS**：时钟源选择

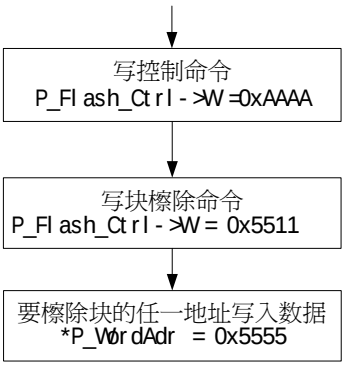
0 ＝ 外部时钟输入/振荡器输入，连接到 XTAL2 脚

1 = 晶体输入，连接到 XTAL1、XTAL2 脚之间

【例 7-1】将第 14 页设置为只读模式

```
P_Flash_RW->W = CW_FlashRW_CMD;                    /* Flash 读写命令*/
P_Flash_RW->B.BK14WENB = CB_BK14WDIS;              /*将第 14 页设为只读 */
```

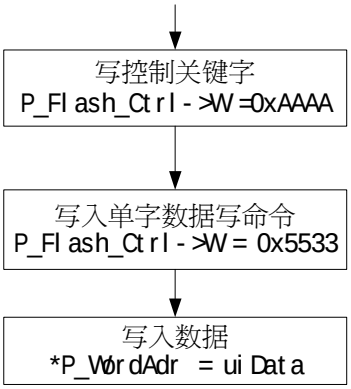
【例 7-2】Flash 块擦除操作




```
//擦除 Flash 的第 14 页的第 0 块

//写控制命令
P_Flash_CMD->W = 0xAAAA;
//写入块擦除命令
P_Flash_CMD->W = 0x5511;
//要檫除的块任一地址写入任意数据
P_WordAdr = (unsigned int *)0xF000;
*P_WordAdr = 0x5555;
```

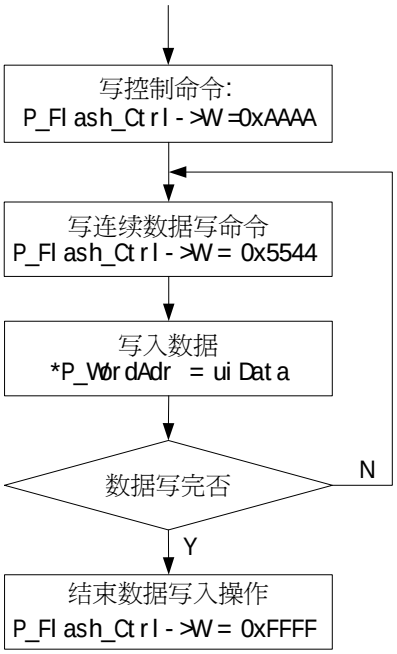
【例 7-3】Flash 单字写操作




```
//写 0x1234 到 0xF000 单元

//写控制命令
P_Flash_CMD->W = 0xAAAA;
//写入单字数据写命令
P_Flash_CMD->W = 0x5533;
//写入数据
P_WordAdr = (unsigned int *)0xF000;
*P_WordAdr = 0x1234;
```

【例 7-4】Flash 连续写操作




```
//写数据到 0xF001-0xF060 的连续单元中
P_WordAdr = (unsigned int *)0xF001;
uiData = 11;
//写控制命令
P_Flash_CMD->W = 0xAAAA;
for(i = 1;i <= 96;i ++)
{
  //写入连续数据写命令
   P_Flash_CMD->W = 0x5544;
   //写入数据
        *P_WordAdr = uiData;
        uiData ++;
        P_WordAdr ++;
}
//结束数据写入操作
P_Flash_CMD->W = 0xFFFF;
```

7.3 Flash 写保护

如果“mass erase”执行，擦除通用区时只能擦除该区的数据，但如果擦除信息区则连同通用区和信息区都可擦除。如果信息区的信息保护选项有效，无法通过 ICE（仿真调试器）或烧录器对 SPMC75X 系列微控制器进行读操作。如果信息区的信息保护选项有效并在仿真使能模式下，ICE 虽不可访问 Flash 的通用区，但可读取信息区的内容。用户唯一可以执行的命令就是“mass erase”，另外，在 ICE 模式下无法访问(读/写)SRAM。细节请参考表 7-3。

表 7-3 各模式下的存贮器访问表

Normal 模式 (ICEN=0)				
	Security =0		Security =1	
	Read	Write	Read	Write
SRAM	Yes	Yes	Yes	Yes
FLASH main block	Yes	Yes	Yes	Yes
FLASH information block	Yes	No	Yes	No
ICE 模式 (ICEN=1)				
SRAM	No	No	Yes	Yes
FLASH main block	No	No(but mass erase)	Yes	Yes
FLASH information block	No	No(but mass erase)	Yes	Yes

在标注操作模式 (ICEN = 0), CPU 访问 flash 数据 和 SRAM. 当在 ICE 模式下，加密有效。ICE 不能对 flash 存储器编程，这样从硬件上防止了非法读取 Falsh 中的程序。图 7-3 为 Flash 的信息保护逻辑。

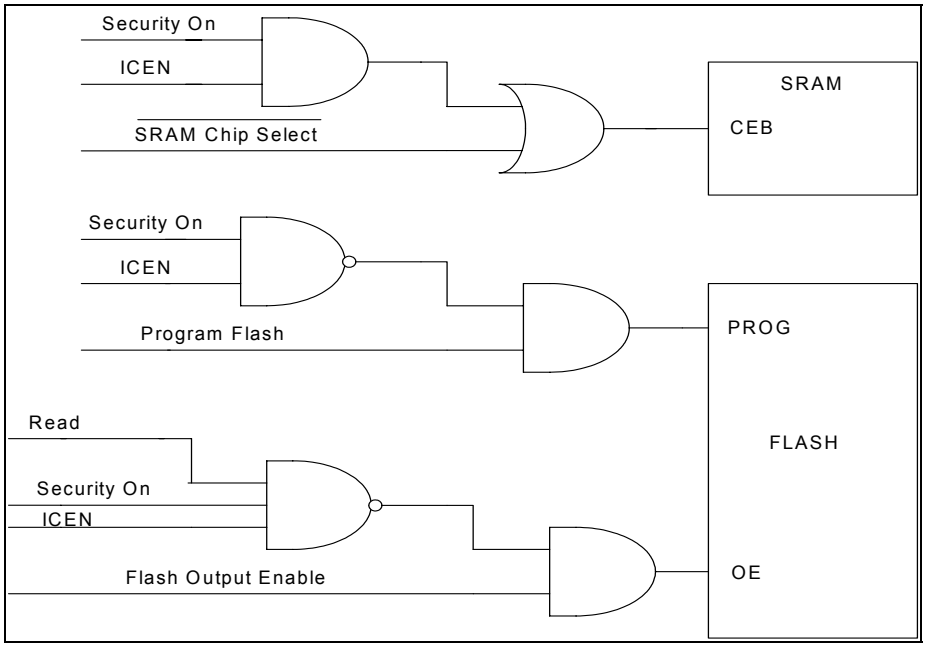


图 7-3　信息保护逻辑框图

第8章 中断

8.1 简介

SPMC75X 系列微控制器具有 38 个中断源。这些中断源可分为 BREAK（软件中断）、FIQ(快速中断请求)和 IRQ0 - IRQ7(普通中断请求)三类。BREAK、FIQ、IRQ 之间的优先级为：BREAK > FIQ > IRQ 0 > IRQ 1 > IRQ 2 > IRQ 3 > IRQ 4 > IRQ 5 >IRQ 6 > IRQ 7。

SPMC75X 系列微控制器支技两种中断模式：普通中断模式和中断嵌套模式。普通中断模式不支持 IRQ 中断嵌套，意即高优先级的 IRQ 中断不能打断低优先级中断服务程序的执行。但 FIQ 和 BREAK 中断可以打断任何 IRQ 中断服务的执行。中断嵌套模式支持 IRQ 中断的嵌套，意即高优先级 IRQ 中断中以打断低优先级 IRQ 中断服务程序的执行。而且支持多级嵌套。在中断嵌套模式中 FIQ 和 BREAK 中断仍有最高优先级可以打断任何 IRQ 中断服务的执行。

注意：BREAK 在任何模式下均有最高优先级，而且它在任何模式下都可以打断其它中断服务的执行，包括 FIQ。

8.2 IRQ 中断向量分配

SPMC75X 系列微控制器的 IRQ 有 IRQ0-IRQ7 共 8 个中断向量。这 8 个中断向量被分配给系统的 35 个中断源。表 8-1 概述了每个 IRQ 中断向量的中断源分配情况。

表 8-1 IRQ 中断分配表

等级	寄存器查询中断标志	名称	描述
IRQ0（最高）	P_INT_Status.FTIF 和 P_Fault1_Ctrl.FTPINIF	FTIN1_INT	故障输入引脚 1 中断
	P_INT_Status.FTIF 和 P_Fault2_Ctrl.FTPINIF	FTIN2_INT	故障输入引脚 2 中断
	P_INT_Status.FTIF 和 P_Fault1_Ctrl.OSF	OS1_INT	输出短路 1 中断
	P_INT_Status.FTIF 和 P_Fault2_Ctrl.OSF	OS2_INT	输出短路 2 中断
	P_INT_Status.OLIF 和 P_OL1_Ctrl.OLIF	OL1_INT	过载引脚 1 中断
	P_INT_Status.OLIF 和 P_OL2_Ctrl.OLIF	OL2_INT	过载引脚 2 中断
	P_INT_Status.OSCSF 和 P_Clk_Ctrl. OSCSF	OSCF_INT	振荡器故障中断
IRQ1	P_INT_Status.PDC0IF 和 P_TMR0_Status.TPRIF	TPR0_INT	定时器 0 TPR 中断
	P_INT_Status.PDC0IF 和 P_TMR0_Status.TGAIF	TGRA0_INT	定时器 0 TGRA 中断
	P_INT_Status.PDC0IF 和 P_TMR0_Status.TGBIF	TGRB0_INT	定时器 0 TGRB 中断
	P_INT_Status.PDC0IF 和 P_TMR0_Status.TGCIF	TGRC0_INT	定时器 0 TGRC 中断
	P_INT_Status.PDC0IF 和 P_TMR0_Status.PDCIF	PDC0_INT	定时器 0 位置改变侦测中断
	P_INT_Status.PDC0IF 和 P_TMR0_Status.TCVIF	TCV0_INT	定时器 0 计数器溢出中断
	P_INT_Status.PDC0IF 和 P_TMR0_Status.TCUIF	TUV0_INT	定时器 0 计数器下溢中断
IRQ2	P_INT_Status.PDC1IF 和 P_TMR1_Status.TPRIF	TPR1_INT	定时器 1 TPR 中断
	P_INT_Status.PDC1IF 和 P_TMR1_Status.TGAIF	TGRA1_INT	定时器 1 TPRA 中断
	P_INT_Status.PDC1IF 和 P_TMR1_Status.TGBIF	TGRB1_INT	定时器 1 TPRB 中断
	P_INT_Status.PDC1IF 和 P_TMR1_Status.TGCIF	TGRC1_INT	定时器 1 TPRC 中断
	P_INT_Status.PDC1IF 和 P_TMR1_Status.PDCIF	PDC1_INT	定时器 1 位置改变侦测中断

等级	寄存器查询中断标志	名称	描述
	P_INT_Status.PDC1IF 和 P_TMR1_Status.TCVIF	TCV1_INT	定时器 1 计数器溢出中断
	P_INT_Status.PDC0IF 和 P_TMR1_Status.TCUIF	TUV1_INT	定时器 1 计数器下溢中断
IRQ3	P_INT_Status.MCP3IF 和 P_TMR3_Status.TPRIF	TPR3_INT	定时器 3 TPR 中断
	P_INT_Status.MCP3IF 和 P_TMR3_Status.TGDIF	TGRD3_INT	定时器 3 TGRD 中断
	P_INT_Status.MCP4IF 和 P_TMR4_Status.TPRIF	TPR4_INT	定时器 4 TPR 中断
	P_INT_Status.MCP4IF 和 P_TMR4_Status.TGDIF	TGRD4_INT	定时器 4 TGRD 中断
IRQ4	P_INT_Status.TPM2IF 和 P_TMR2_Status.TPRIF	TPR2_INT	定时器 2 TPR 中断
	P_INT_Status.TPM2IF 和 P_TMR2_Status.TGAIF	TGRA2_INT	定时器 2 TGRA 中断
	P_INT_Status.TPM2IF 和 P_TMR2_Status.TGBIF	TGRB2_INT	定时器 2 TGRB 中断
IRQ5	P_INT_Status.EXT0IF	EXT0_INT	外部中断 0
	P_INT_Status.EXT1IF	EXT1_INT	外部中断 1
IRQ6	P_INT_Status.UARTIF 和 P_UART_Status.RXIF	UART_RX_INT	UART 接受完成中断
	P_INT_Status.UARTIF 和 P_UART_Status.TXIF	UART_TX_INT	UART 发送就绪中断
	P_INT_Status.SPIIF 和 P_SPI_RxStatus.SPIRXIF	SPI_RX_INT	SPI 接受中断
	P_INT_Status.SPIIF 和 P_SPI_TxStatus.SPITXIF	SPI_TX_INT	SPI 发送中断
IRQ7 (最低)	P_INT_Status.KEYIF	IOKEY_INT	IO 按键唤醒中断
	P_INT_Status.ADCIF 和 P_ADC_Ctrl.ADCIF	ADC_INT	模/数转换完成中断
	P_INT_Status.CMTIF 和 P_CMT_Ctrl.CM0IF	CMT0_INT	比较匹配定时器 0 中断
	P_INT_Status.CMTIF 和 P_CMT_Ctrl.CM1IF	CMT1_INT	比较匹配定时器 1 中断

8.3 控制寄存器

SPMC75X 系列微控制器的中断控制模块共有 3 个控制寄存器，如表 8-2 所示。通过这 3 个控制寄存器可以完成中断控制模块所有功能的控制。

表 8-2 中断控制模块的控制寄存器

地址	寄存器	名称
70A0h	P_INT_Status	中断状态标志寄存器
70A4h	P_INT_Priority	IRQ与FIQ优先级选择寄存器
70A8h	P_MisINT_Ctrl	综合中断控制寄存器

P_INT_Status ($70A0)：中断状态标志寄存器

该寄存器标识了按键唤醒，UART、SPI、EXT1、EXT0、模-数转换器、TMR4、TMR3、TMR2、TMR1、TMR0、比较匹配定时器、振荡器状态的中断源的中断状态，通过它可查询是否发生了相应的中断事件。

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
KEYIF	UARTIF	SPIIF	EXT1IF	EXT0IF	ADCIF	MCP4IF	MCP3IF

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
TPM2IF	PDC1IF	PDC0IF	CMTIF	保留	OLIF	OSCSF	FTIF

第 15 位　　KEYIF：按键唤醒中断状态标志。此标志可指出按键唤醒中断是否发生，写入‘1’可清除此标志

读出：

0 ＝ 未发生

1 ＝ 已发生

第 14 位　　UARTIF：UART 中断状态标志。此标志可指出 UART 中断是否发生

读出：

0 ＝ 未发生

1 ＝ 已发生

第 13 位　　SPIIF：SPI 中断状态标志。此标志可指出 SPI 中断是否发生

读出：

0 ＝ 未发生

1 ＝ 已发生

第 12 位　　EXT1IF：外部中断 1 的状态标志。此标志可指出外部中断 1 是否发生，写入‘1’可清除此标志

读出：

0 ＝ 未发生

1 ＝ 已发生

第 11 位　　EXT0IF：外部中断 0 的状态标志。此标志可指出外部中断 0 是否发生，写入‘1’可清除此标志

读出：

0 ＝ 未发生

1 ＝ 已发生

第 10 位　　ADCIF：模/数转换器中断的状态标志。此标志可指出模/数转换完成中断是否发生

读出：

0 ＝ 未发生

1 ＝ 已发生

此位也储存在 ADC 模块中。

第 9 位　　MCP4IF：MCP 定时器 4 的中断的状态标志。此标志可指出 MCP 定时器 4

的中断是否发生

读出：

0 = 未发生 MCP 定时器 4 的中断

1 = 已发生 MCP 定时器 4 的中断

第 8 位 **MCP3IF**： 定时器/PWM 模块 MCP 定时器 3 的中断的状态标志。此标志可指出 MCP 定时器 3 的中断是否发生

读出：

0 = 未发生 MCP 定时器 3 的中断

1 = 已发生 MCP 定时器 3 的中断

第 7 位 **TPM2IF**： 定时器/PWM 模块 TPM 定时器 2 的中断的状态标志。此标志可指出 TPM 定时器 2 的中断是否发生

读出：

0 = 未发生 TPM 定时器 2 的中断

1 = 已发生 TPM 定时器 2 的中断

第 6 位 **PDC1IF**： PDC 定时器 1 的中断的状态标志。此标志可指出 PDC 定时器 1 的中断是否发生

读出：

0 = 未发生 PDC 定时器 1 的中断

1 = 已发生 PDC 定时器 1 的中断

第 5 位 **PDC0IF**：定时器 0 的中断的状态标志。此标志可指出 PDC 定时器 0 的中断是否发生

读出：

0 = 未发生 PDC 定时器 0 的中断

1 = 已发生 PDC 定时器 0 的中断

第 4 位 **CMTIF**： 比较匹配定时器(CMT)中断的状态标志。此标志可指出比较匹配定时器中断是否发生

读出：

0 = 未发生 CMT 中断

1 = 已发生 CMT 中断

第 3 位 保留

第 2 位 **OLIF**： 过载中断的状态标志。此标志可指出是否发生过载

读出：

0 = 未发生 CMT 中断

1 = 已发生 CMT 中断

第 1 位 **OSCSF**：振荡器的状态标志。此标志可指出芯片内晶体振荡器或外部输入时钟是否正常运行。如果振荡器或外部时钟输入停止，此位被置位‘1’

读出：

0 ＝ 振荡器运行正常

1 ＝ 振荡器故障

将此位写为‘1’可清除此标志

第 0 位　　FTIF：故障保护中断的状态标志。此标志可指出故障保护中断是否发生。共有 4 个可能的中断源：FTINP1、FTINP2、TIO3A～F 输出短路、TIO4A～F 输出短路。如果以上 4 个中断标志中的任何一个被置为‘1’，此标志的读取值为‘1’

读出：

0 ＝ 未发生

1 ＝ 已发生

P_INT_Priority ($70A4)： IRQ 与 FIQ 优先级选择寄存器

该寄存器用来设置各外设中断源的优先级，可将中断源设为 IRQ（普通中断）或 FIQ（快速中断）。默认中断等级为 IRQ。注：在 P_INT_Priority 中，同时只能有一个中断源被设为 FIQ。

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
KEYIP	UARTIP	SPIIP	保留	EXTIP	ADCIP	MCP4IP	MCP3IP

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R	R	R/W	R/W
0	0	0	0	0	0	0	0
TPM2IP	PDC1IP	PDC0IP	CMTIP	保留	OLIP	OSCIP	FTIP

第 15 位　　KEYIP：按键唤醒中断优先权选择位。此位可以选择按键唤醒中断优先权

0 ＝ IRQ7

1 ＝ FIQ

第 14 位　　UARTIP：UART 中断优先权选择位。此位可以选择 UART 中断优先权

0 ＝ IRQ7

1 ＝ FIQ

第 13 位　　SPIIP：SPI 中断优先权选择位。此位可以选择 SPI 中断优先权

0 ＝ IRQ6

1 ＝ FIQ

第 12 位　　保留

第 11 位　　EXTIP：外部中断优先权选择位。此位可以选择外部中断优先权

0 ＝ IRQ5

1 = FIQ

第 10 位 **ADCIP**：ADC 中断优先权选择位。此位可以选择 ADC 中断优先权

0 = IRQ7

1 = FIQ

第 9 位 **MCP4IP**：MCP 定时器 4 的中断优先权选择位。此位可以选择 MCP 定时器 4 的中断优先权

0 = IRQ3

1 = FIQ

第 8 位 **MCP3IP**：MCP 定时器 3 的中断优先权选择位。此位可以选择 MCP 定时器 3 的中断优先权

0 = IRQ3

1 = FIQ

第 7 位 **TPM2IP**：TPM 定时器 2 的中断优先权选择位。此位可以选择 TPM 定时器 2 的中断优先权

0 = IRQ4

1 = FIQ

第 6 位 **PDC1IP**：PDC 定时器 1 的中断优先权选择位。此位可以选择 PDC 定时器 1 的中断优先权

0 = IRQ2

1 = FIQ

第 5 位 **PDC0IP**：PDC 定时器 0 的中断优先权选择位。此位可以选择 PDC 定时器 0 的中断优先权

0 = IRQ1

1 = FIQ

第 4 位 **CMTIP**：CMT 中断优先权选择位。此位可以选择 CMT 中断优先权

0 = IRQ7

1 = FIQ

第 3 位 保留

第 2 位 **OLIP**：过载中断优先权选择位。此位可以选择过载中断优先权

0 = IRQ0

1 = FIQ

第 1 位 **OSCIP**：振荡器故障中断优先权选择位。此位可以选择振荡器故障中断优先权

0 = IRQ0

1 = FIQ

第 0 位 **FTIP**：故障保护中断优先权选择位。此位可以选择故障保护中断优先权

0 = IRQ0

1 = FIQ

P_MisINT_Ctrl (\$70A8)：综合中断控制寄存器

该寄存器用来使能按键唤醒中断、外部中断 0 和外部中断 1，同时，这该寄存器还用作设置外部中断的触发边沿。

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R	R	R
0	0	0	0	0	0	0	0
KEYIE	EXT1MS	EXT0MS	EXT1IE	EXT0IE	保留		

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

第 15 位　KEYIE：按键唤醒中断允许位

写入：

0 ＝ 禁止

1 ＝ 允许

第 14 位　EXT1MS：外部中断 1 触发器边沿选择位。此位可以选择 EXINT1 的触发器边沿

0 ＝ 上升沿触发器

1 ＝ 下降沿触发器

第 13 位　EXT0MS：外部中断 0 边沿触发器选择位。此位可以选择 EXINT1 的触发器边沿

0 ＝ 上升沿触发

1 ＝ 下降沿触发

第 12 位　EXT1IE：外部中断 1 的允许位。外部中断输入 EXINT1 按 FCK/2 时钟频率取样。任何小于 4 倍取样时钟频率的脉冲都会被忽略

0 ＝ 禁止

1 ＝ 允许

第 11 位　EXT0IE：外部中断 0 的允许位。外部中断输入 EXINT0 按 FCK/2 时钟频率取样。任何小于 4 倍取样时钟频率的脉冲都会被忽略

0 ＝ 禁止

1 ＝ 允许

第 10:0 位　保留

8.4 中断模型

在中断过程中，返回的 PC 值与 SR 寄存器(状态寄存器)由 CPU 自动保存到堆栈。典型用法是，用户应该在中断发生时保存关键寄存器的值，如 R1～R5 寄存器。这必须要由软件编程来实现。保存信息的动作通常被称为“入栈”，而在中断返回前的恢复动作通常为“出栈”。任何中断都应由“RETI”指令返回，否则堆栈溢出，与数据区重叠，系统会发生崩溃。

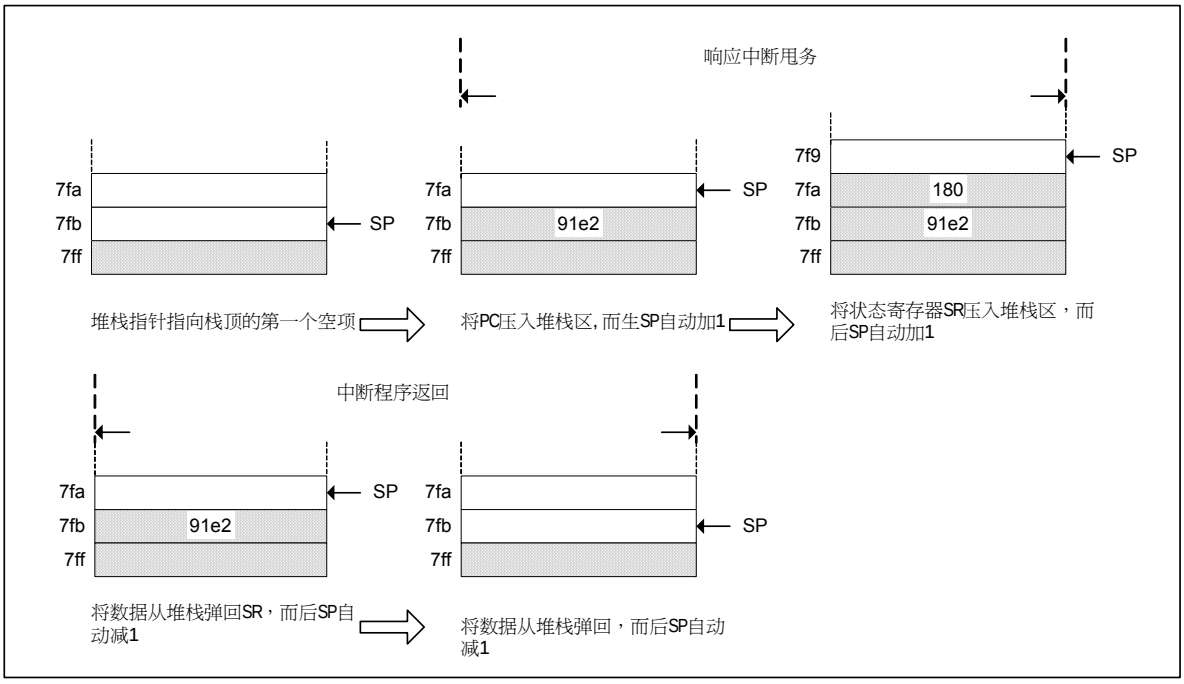


图 8-1 中断时的堆栈响应

```
//保护和恢复 R1～R5 寄存器
        push    r1,r5    to [sp]
                ……
        pop r1,r5    from [sp]
        reti
```

下面的指令给出了 SPMC75X 系列微控制器中断服务程序的框架结构。中断入口的名称由编译器保留,请不要更改入口的名称。

(A) 使用汇编语言

```
//*********************************************************************
// Function: Fast Interrupt Service routine Area
//   Service for  (1) FIQ
//                (2) IRQ 0 ～ IRQ 7
//  User's FIQ must hook on here
//  _FIQ:              //快速中断入口
//  _IRQ1:             //中断入口
```

```
//  _IRQ2:              //中断入口
//  _IRQ3:              //中断入口
//  _IRQ4:              //中断入口
//  _IRQ5:              //中断入口
//  _IRQ6:              //中断入口
//  _IRQ7:              //中断入口
//*********************************************************************
.include SPMC75F2413A.inc       // 包含寄存器定义和常数定义
.TEXT
.public _BREAK;
.public _FIQ;
.public  _IRQ0,_IRQ1, _IRQ2,_IRQ3,_IRQ4, _IRQ5,_IRQ6, _IRQ7
//=================================================
//功能：中断服务子程序区域
// FIQ 服务子程序
//=================================================
_FIQ:
         push r1,r5 to [sp];
         //-----------------------------------------------
         //Add FIQ Function
         //-----------------------------------------------
         pop r1,r5 from [sp];
     reti;

//=================================================
//功能：中断服务子程序区域
//IRQ1 - IRQ7 服务子程序
//用户的 IRQ 必须在这里调用
//=================================================
_BREAK:
     push r1,r5 to [sp];
     //-----------------------------------------------
     //加入中断服务程序
     //-----------------------------------------------
     pop r1,r5 from [sp];
     reti;
//=================================================
```

```
_IRQ0:
        push r1,r5 to [sp];
        //-----------------------------------------------
    // Add FTINT function
        //-----------------------------------------------
        pop r1,r5 from [sp];
     reti;

//================================================
_IRQ1:
        push r1,r5 to [sp];
        //-----------------------------------------------
        //加入 PDC0 中断函数

        //-----------------------------------------------
        pop r1,r5 from [sp];
     reti;
//================================================
_IRQ2:
        push r1,r5 to [sp];
        //-----------------------------------------------
        //加入 PDC1 中断函数

        //-----------------------------------------------
        pop r1,r5 from [sp];
     reti;

//================================================
_IRQ3:
        push r1,r5 to [sp];
        //-----------------------------------------------
        //加入 MCP3 和 MCP4 中断函数

        //-----------------------------------------------
        pop r1,r5 from [sp];
     reti;
```

```
//================================================
_IRQ4:
        push r1,r5 to [sp];
        //------------------------------------------------
        //加入 TPM2 中断函数
        //------------------------------------------------
        pop r1,r5 from [sp];
      reti;

//================================================
_IRQ5:
        push r1,r5 to [sp];
        //------------------------------------------------
        //加入外部中断(EXINT)函数

        //------------------------------------------------
        pop r1,r5 from [sp];
      reti;

//================================================
_IRQ6:
        push r1,r5 to [sp];
        //------------------------------------------------
        //加入 UART 和 SPI 函数

        //------------------------------------------------
        pop r1,r5 from [sp];
     reti;

//================================================
_IRQ7:
        push r1,r5 to [sp];
        //------------------------------------------------
        //加入 KEYINT, ADCINT, CMTINT 函数

        //------------------------------------------------
        pop r1,r5 from [sp];
      reti;
```

```
//================================================
// End of isr.asm
//================================================
```

(B) 使用 C 语言

```
void BREAK(void) __attribute__ ((ISR));
void BREAK(void)
{
}
void FIQ(void) __attribute__ ((ISR));
void FIQ(void)
{
}
void IRQ0(void) __attribute__ ((ISR));
void IRQ0(void)
{
}
void IRQ1(void) __attribute__ ((ISR));
void IRQ1(void)
{
}
void IRQ2(void) __attribute__ ((ISR));
void IRQ2(void)
{
}
void IRQ3(void) __attribute__ ((ISR));
void IRQ3(void)
{
}
void IRQ4(void) __attribute__ ((ISR));
void IRQ4(void)
{
}
void IRQ5(void) __attribute__ ((ISR));
void IRQ5(void)
{
```

```
}
void IRQ6(void) __attribute__ ((ISR));
void IRQ6(void)
{

}
void IRQ7(void) __attribute__ ((ISR));
void IRQ7(void)
{
}
```

8.5 设计参考

当使用中断源时，必须遵循下面给出的步骤：

i. 通过 P_MisINT_Ctrl 控制寄存器或外设控制寄存器允许中断

ii. 设置 P_INT_Priority ($7061)，以对中断的 IRQ 与 FIQ 选择进行配置

iii. 使用下列指令来允许 IRQ 和 FIQ

a. IRQ on //打开 IRQ 中断

b. INT IRQ //允许 IRQ 中断，禁止 FIQ

c. INT FIQ //允许 FIQ 中断，禁止 IRQ

d. INT FIQ, IRQ //同时允许 IRQ 和 FIQ 中断

【例 8-1】按键唤醒中断

```
/* 将 IOA[15]设置键唤醒中断源并使能该中断 */
 P_IOA_SPE->W      = 0x0000;                    /* 禁止 IOA 特殊功能 */
 P_IOA_Dir->W = 0x0000;                         /* 置为输入 */
 P_IOA_Attrib->W = 0x0000;                      /* 非悬浮输入 */
 P_IOA_Buffer->W = 0x0000;                      /* 上拉  */
 P_IOA_KCER->B.KC15EN = 1;                      /* 设置键唤醒源 */
 P_MisINT_Ctrl->B.KEYIE  = 1;                   /* 使能键唤醒中断 */
 Data =     P_IOA_Latch->W;
 INT_IRQ();
void IRQ7(void) __attribute__ ((ISR));
void IRQ7(void)
{
 if(P_INT_Status->B.KEYIF)                      /* 清除键唤醒状态标志 */
       P_INT_Status->B.KEYIF  = 1;
}
```

例 8-1 按键唤醒中断设计参考

【例 8-2】外部中断

```
/* 将 IOC[2] 设置为外部中断源并使能该中断*/
 P_IOC_SPE->B.EXINT0EN = 1;                    /* 使能外部中断 0 引脚 */
 P_MisINT_Ctrl->B.EXT0IE =    1;               /*使能外部中断 0 */
 INT_IRQ();

void IRQ5(void) __attribute__ ((ISR));
void IRQ5(void)
{
if(P_INT_Status->B.EXT0IF)
P_INT_Status->B.EXT0IF = 1;
}
```

例 8-2 外部中断设计参考

【例 8-3】 按键唤醒中断

```
/* 将 IOA[15]设置键唤醒中断源并使能该中断 */
 P_IOA_SPE->W      = 0x0000;                   /* 禁止 IOA 特殊功能 */
 P_IOA_Dir->W = 0x0000;                        /* 置为输入 */
 P_IOA_Attrib->W = 0x0000;                     /* 非悬浮输入 */
 P_IOA_Buffer->W = 0x0000;                     /* 上拉  */
 P_IOA_KCER->B.KC15EN = 1;                     /* 设置键唤醒源 */
 P_MisINT_Ctrl->B.KEYIE  = 1;                  /* 使能键唤醒中断 */
 Data =      P_IOA_Latch->W;
 INT_IRQ();

void IRQ7(void) __attribute__ ((ISR));
void IRQ7(void)
{
 if(P_INT_Status->B.KEYIF)                     /* 清除键唤醒状态标志 */
       P_INT_Status->B.KEYIF  = 1;
}
```

例 8-3 按键唤醒中断设计参考

【例 8-4】外部中断

```
/* 将 IOC[2] 设置为外部中断源并使能该中断*/
 P_IOC_SPE->B.EXINT0EN = 1;                    /* 使能外部中断 0 引脚 */
 P_MisINT_Ctrl->B.EXT0IE =    1;               /*使能外部中断 0 */
 INT_IRQ();

void IRQ5(void) __attribute__ ((ISR));
void IRQ5(void)
{
if(P_INT_Status->B.EXT0IF)
P_INT_Status->B.EXT0IF = 1;
```

```
}
```

例 8-4 外部中断设计参考

第9章 I/O 端口

9.1 简介

SPMC75X 系列微控制器共有 4 个通用 IO 端口：IOA、IOB、IOC 和 IOD，每个 16 位。每个 I/O 引脚都可通过软件编程进行逐位配置。除端口 D 外，其他端口的 I/O 引脚都可通过编程来实现特殊功能。换句话说，这些 I/O 端口与许多功能控制信号是复用的，例如端口 A[15:8]就可提供唤醒功能，并可从低功耗模式按键唤醒。

SPMC75X 系列微控制器 I/O 口的特殊功能寄存器是通过设置相应的特殊功能寄存器来实现的。当特殊功能有效时，通用 IO 功能即被禁用。此外，一些特殊功能对 I/O 引脚的功能设置有特殊要求。例如 AD 转换输入引脚和 SPI 接口。此时，I/O 的方向与属性寄存器应设置为特定的状态。

9.2 I/O 口结构

SPMC75X 系列微控制器的 IO 结构如图 9-1 所示。SPMC75X 系列微控制器的 IO 端口主要有 5 个控制寄存器控制：数据、缓冲器、方向、 属性、 锁存和特殊功能寄存器。寄存器的命名规则如下所示：

数据寄存器	→	P_IOx_Data
缓冲寄存器	→	P_IOx_Buffer
数据方向寄存器	→	P_IOx_Dir
属性寄存器	→	P_IOx_Attrib
锁存寄存器	→	P_IOA_Latch
特殊功能寄存器	→	P_IOA_SPE, P_IOB_SPE, P_IOC_SPE

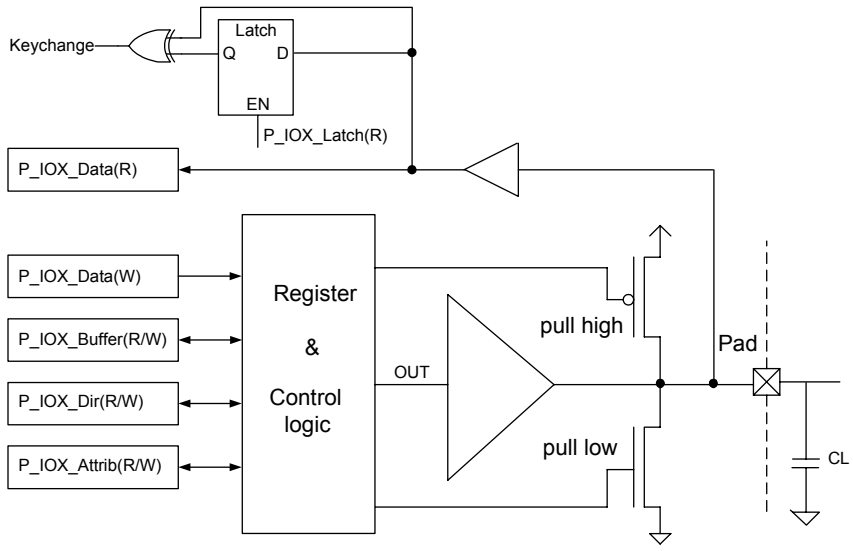


图 9-1 IO 结构图

SPMC75X 系列微控制器的普通 I/O 功能是由 P_IOx_Data、P_IOx_Dir 和 P_IOx_Attrib 这几个控制寄存器来设置的。规则如下，详见表 9-1 所示：

a. 方向寄存器 — 决定引脚是输入还是输出。

b. 属性寄存器 — 决定引脚的属性：输入为悬浮或上拉/下拉；输出是否为反向。

c. 数据寄存器 — 决定引脚的初始内容。对于输入，数据设定也决定了上拉或下拉的设定。例如：设端口 A.0 为下拉输入，则端口 A 的方向、属性和数据控制端口中的第 0 位都应被设为 0。如果端口 A.1 为悬浮输入，而且有唤醒功能，则端口 A 的方向、属性和数据控制端口中的第 1 位应相应地赋值为“010”。注意每个端口都具有 16 个方向、属性和数据位。用户在配置 I/O 时应特别注意。

表 9-1 I/O 功能配置表

方向	属性	数据	功能	唤醒	描述
0	0	0	下拉*	是**	带下拉电阻的输入
0	0	1	上拉	是**	带上拉电阻的输入
0	1	0	悬浮	是**	悬浮的输入
0	1	1	悬浮	否	悬浮的输入
1	0	0	反向	否	数据反向输出(在数据端口写入“0”，则在 I/O 端口输出“1”)
1	0	1	反向	否	数据反向输出(在数据端口写入“1”，则在 I/O 端口输出“0”)
1	1	0	非反向	否	缓冲输出(数据不反向)
1	1	1	非反向	否	缓冲输出(数据不反向)

* 默认为下拉输入。

**只有端口 A[15:8]的状态配置为“000”， “001”和“010” 才有唤醒能力。

9.2.1 按键唤醒中断

IOA[15..8]具有键唤醒功能，如果端口其中的任何一个 I/O 状态异于锁存寄存器 P_IOA_Latch 的内容即发生键唤醒事件。IOA[15:8]有专用的噪声滤波器，以防止 IO 管脚的噪声干扰引起键唤醒电路的误动作。图 9-3 是一个键唤醒功能的结构框图,图 9-4 是系统响应键唤醒中断的时序图

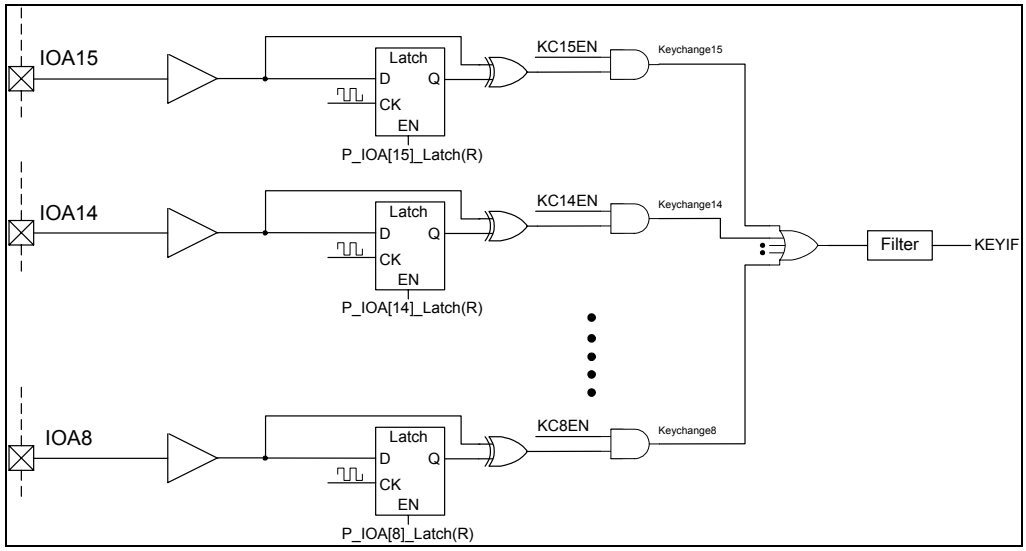


图 9-3　键唤醒结构框图

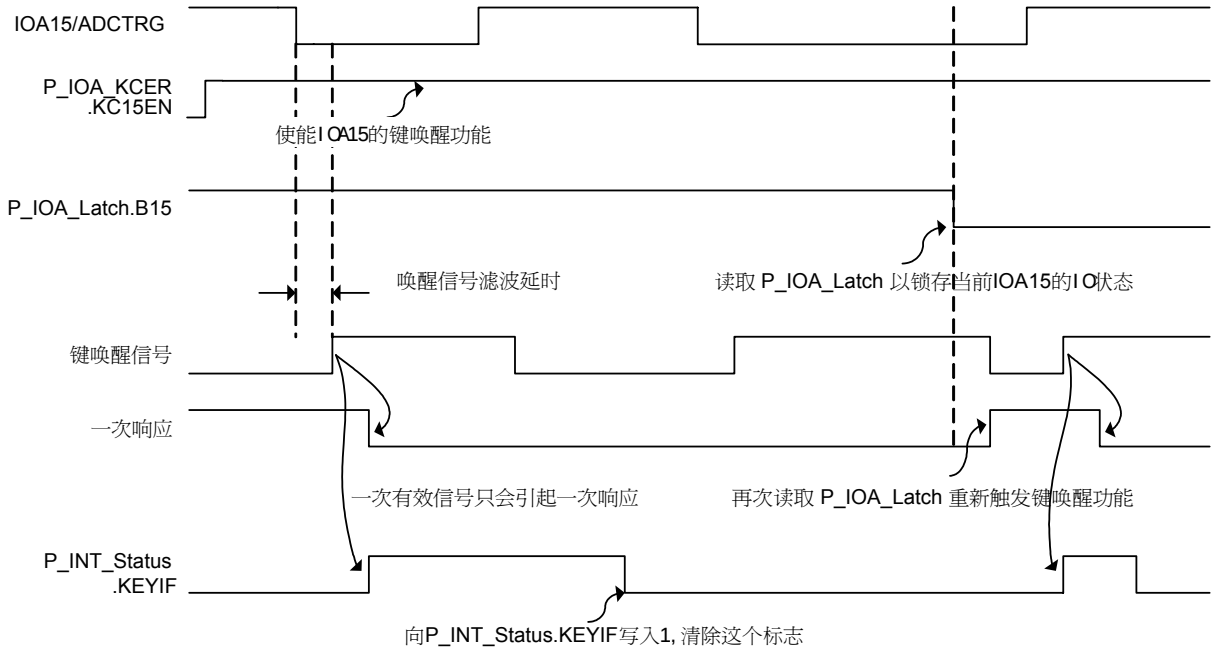


图 9-4　键唤醒时序图

9.3 IO 端口 A

端口 A 是一个多功能复用 IO 端口。端口 A 除了拥有普通的 I/O 功能外，还具有键中断输入、外部计数时钟输入、PWM 输出和捕获输入功能。端口 A 的所功能都通过 6 个控制寄存器切换控制。

P_IOA_Data ($7060H)：IO 端口 A 的数据寄存器

向 P_IOA_Data 写入数据是写入 IO 的输出缓冲寄存器，而读取则是读管脚状态。

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Data							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Data							

P_IOA_Buffer ($7061H)：IO 端口 A 缓冲寄存器

P_IOA_Buffer 写入与 P_IOA_Data 写入相同。而读取则是读 IO 的输出缓冲寄存器，利用这一特性，可实现单个 I/O 口的独立控制。

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Buffer							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Buffer							

注：读 P_IOA_Data 和读 P_IOA_Buffer 从不同的物理路径读取数据。读 P_IOA_Data 时的内容来自 I/O 端口 A 的引脚电平,而读 P_IOA_Buffer 内容则来自 I/O 端口 A 的输出缓冲寄存器。

P_IOA_Dir ($7062H)：IO 端口 A 方向寄存器

端口 A 各 I/O 的方向设定寄存器。当相应的位为“1”时，I/O 被设为输出状态；当相应的位为“0”时，I/O 被设为输入状态。

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Dir							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

0	0	0	0	0	0	0	0
P_IOA_Dir							

P_IOA_Attrib ($7063H)：IO 端口 A 属性寄存器

端口 A 各 I/O 的属性设定寄存器。它同 P_IOA_Dir 和 P_IOA_Data 相互配合可以将 I/O 设为多种属性。详情请见表 9-1

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
P_IOA_Attrib							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
P_IOA_Attrib							

P_IOA_Latch ($7064H)：IO 端口 A 锁存寄存器

P_IOA_Latch 是为键唤醒功能服务的专用寄存器。在使用键唤醒中断时，必需读取 P_IOA_Latch，以将当前的 I/O 状态保存。当键唤醒功能的 I/O 状态发生改变（与读取 P_IOA_Latch 时锁存的数据不相符）时，就会置键中断标志，以标识这一事件的发生。

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
P_IOA_Latch							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

注：此外，IOA[15:8]是按键唤醒功能的信号源。为激活按键唤醒功能，必须读取 P_IOA_Latch(R)($7064H)，以锁存 I/O 端口 A 的状态，而且必须在进入 Standby 模式之前使能按键唤醒功能。当监测到端口 A 的 I/O 状态发生变化时，唤醒功能被触发。

P_IOA_SPE ($7080H)：IO 端口 A 特殊功能寄存器

端口 A 的特殊功能使能寄存器。用来使能 IOA[14..9] 的特殊功能。相应的位功能定义如下所述。

B15	B14	B13	B12	B11	B10	B9	B8
R	R/W	R/W	R/W	R/W	R/W	R/W	R

0	0	0	0	0	0	0	0
保留	TCLKDEN	TCLKCEN	TCLKBEN	TCLKAEN	TIO2BEN	TIO2AEN	保留

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

第 15　　保留

第 14 位　　TCLKDEN：外部时钟 D 的输入引脚(通道 1 相位计数模式 D 相位输入) 使能

0 ＝ 禁止

1 ＝ 使能

第 13 位　　TCLKCEN：外部时钟 C 的输入引脚(通道 1 相位计数模式 C 相位输入) 使能

0 ＝ 禁止

1 ＝ 使能

第 12 位　　TCLKBEN：外部时钟 B 的输入引脚(通道 1 相位计数模式 B 相位输入) 使能

0 ＝ 禁止

1 ＝ 使能

第 11 位　　TCLKAEN：外部时钟 A 的输入引脚(通道 1 相位计数模式 A 相位输入) 使能

0 ＝ 禁止

1 ＝ 使能

第 10 位　　TIO2BEN：P_TMR2_TGRB 捕获输入/PWM 输出使能

0 ＝ 禁止

1 ＝ 使能

第 9 位　　TIO2AEN：P_TMR2_TGRA 输入捕获/PWM 输出使能

0 ＝ 禁止

1 ＝ 使能

第 8:0 位　　保留

P_IOA_KCER ($7084H)：IO 端口 A 按键唤醒寄存器

P_IOA_KCER 使能 IOA[15..8]的键中断功能。

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
KC15EN	KC14EN	KC13EN	KC12EN	KC11EN	KC10EN	KC9EN	KC8EN

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R

0	0	0	0	0	0	0	0
保留							

第 15 位　　KC15EN：端口 A.15 按键唤醒使能

0 = 禁止

1 = 使能

第 14 位　　KC14EN：端口 A.14 按键唤醒使能

0 = 禁止

1 = 使能

第 13 位　　KC13EN：端口 A.13 按键唤醒使能

0 = 禁止

1 = 使能

第 12 位　　KC12EN：端口 A.12 按键唤醒使能

0 = 禁止

1 = 使能

第 11 位　　KC11EN：端口 A.11 按键唤醒使能

0 = 禁止

1 = 使能

第 10 位　　KC10EN：端口 A.10 按键唤醒使能

0 = 禁止

1 = 使能

第 9 位　　KC9EN：端口 A.9 按键唤醒使能

0 = 禁止

1 = 使能

第 8 位　　KC8EN：端口 A.8 按键唤醒使能

0 = 禁止

1 = 使能

第 7:0 位　　保留

端口 A 的特殊功能

端口 A 有以下复用特殊功能：

1. 模/数转换接口 → Port A [7:0] A[15]
2. 外部时钟输入 → Port A [14:11]
3. TMR2 CCP 引脚 → Poet A[10:9]

每个特殊功能都有一个使能控制信号，用来将指定的引脚配置成所需的属性。表 9-3 给出了控制信号和对特殊功能的描述。

表 9-3 端口 A 的特殊功能表

	SFR 引脚	类型	PHB	PL	OEB	描述
IOA15	ADCETRG	I	User	User	ADCEXTRIGEN	A/D 转换的外部触发输入
IOA14	TCLKD	I	User	User	TCLKDEN	外部时钟 D 的输入引脚（定时器 1 相位计数模式 D 相位输入）
IOA13	TCLKC	I	User	User	TCLKCEN	外部时钟 C 的输入引脚（定时器 1 相位计数模式 C 相位输入）
IOA12	TCLKB	I	User	User	TCLKBEN	外部时钟 B 的输入引脚（定时器 0 相位计数模式 B 相位输入）
IOA11	TCLKA	I	User	User	TCLKAEN	外部时钟 A 的输入引脚（定时器 0 相位计数模式 A 相位输入）
IOA10	TIO2B	User	1	0	TIO2BEN	TGRB_2 捕获输入/PWM 输出引脚 B
IOA9	TIO2A	User	1	0	TIO2AEN	TGRA_2 输入捕获/PWM 输出引脚 A
IOA8	–	–	–	–	–	–
IOA7	ADCCH7	I	1	0	ADCI7EN	ADC 通道 7 的模拟输入
IOA6	ADCCH6	I	1	0	ADCI6EN	ADC 通道 6 的模拟输入
IOA5	ADCCH5	I	1	0	ADCI5EN	ADC 通道 5 的模拟输入
IOA4	ADCCH4	I	1	0	ADCI4EN	ADC 通道 4 的模拟输入
IOA3	ADCCH3	I	1	0	ADCI3EN	ADC 通道 3 的模拟输入
IOA2	ADCCH2	I	1	0	ADCI2EN	ADC 通道 2 的模拟输入
IOA1	ADCCH1	I	1	0	ADCI1EN	ADC 通道 1 的模拟输入
IOA0	ADCCH0	I	1	0	ADCI0EN	ADC 通道 0 的模拟输入

9.4 IO 端口 B

端口 B 是一个多功能复用 IO 端口。端口 B 除了拥有普通的 I/O 功能外，还具有 PWM 输出、捕获输入功能和串行通信接口的功能。端口 B 的所功能都通过 5 个控制寄存器切换控制。

P_IOB_Data ($7068H)：IO 端口 B 的数据寄存器

向 P_IOB_Data 写入数据是写入 IO 的输出缓冲寄存器，而读取则是读管脚状态。

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Data							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Data							

P_IOB_Buffer ($7069H)： IO 端口 B 缓冲寄存器

P_IOB_Buffer 写入与 P_IOB_Data 写入相同。而读取则是读 IO 的输出缓冲寄存器，利用这一特性，可实现单个 I/O 口的独立控制。

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Buffer							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Buffer							

注：读 P_IOB_Data 和读 P_IOB_Buffer 从不同的物理路径读取数据。读 P_IOB_Data 时的内容来自 I/O 端口 B 的引脚电平,而读 P_IOB_Buffer 内容则来自 I/O 端口 B 的输出缓冲寄存器。

P_IOB_Dir ($706AH)：IO 端口的方向寄存器

端口 B 各 I/O 的方向设定寄存器。当相应的位为“1”时，I/O 被设为输出状态；当相应的位为“0”时，I/O 被设为输入状态。

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Dir							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Dir							

P_IOB_Attrib ($706BH)：IO 端口 B 的属性寄存器

端口 B 各 I/O 的属性设定寄存器。它同 P_IOB_Dir 和 P_IOB_Data 相互配合可以将 I/O 设为多种属性。详见表 9-1

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
P_IOB_Attrib							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1

P_IOB_Attrib

P_IOB_SPE ($7081H)： IO 端口 B 的特殊功能寄存器

端口 B 的特殊功能使能寄存器。用来使能 IOB[10..0]的特殊功能。相应的位功能定义如下所述。

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
保留					TIO0AEN	TIO0BEN	TIO0CEN

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	1	1	1	1	1	1
OL1EN	FTIN1EN	U1EN	V1EN	W1EN	U1NEN	V1NEN	W1NEN

第 15:11　保留

第 10 位　**TIO0AEN**：P_TMR0_TGRA 捕获输入/PWM 输出引脚或位置侦测输入使能

0 ＝ 禁止

1 ＝ 使能

第 9 位　**TIO0BEN**：P_TMR0_TGRB 输入捕获/PWM 输出引脚或位置侦测输入使能

0 ＝ 禁止

1 ＝ 使能

第 8 位　**TIO0CEN**：P_TMR0_TGRC 输入捕获/PWM 输出引脚或位置侦测输入使能

0 ＝ 禁止

1 ＝ 使能

第 7 位　**OL1EN**：过载保护输入 1 使能

0 ＝ 禁止

1 ＝ 使能

第 6 位　**FTIN1EN**：外部故障保护输入 1 使能

0 ＝ 禁止

1 ＝ 使能

第 5 位　**U1EN**：U1 模式选择

0 ＝ GPIO

1 ＝ U1 相

第 4 位　**V1EN**：V1 模式选择

0 ＝ GPIO

1 ＝ V1 相

第 3 位　**W1EN**：W1 模式选择

0 ＝ GPIO

1 = W1 相

第 2 位 **U1NEN**：U1N 模式选择

0 = GPIO

1 = U1N 相

第 1 位 **V1NEN**：V1N 模式选择

0 = GPIO

1 = V1N 相

第 0 位 **W1NEN**：W1N 引式选择

0 = GPIO

1 = W1N 相

注 : P_IOB_SPE[5 :0]默认 '1'

端口 B 的特殊功能

端口 B 有以下复用的特殊功能：

1. SPI 接口/UART1 接口 → Port B [13:11]
2. 捕获/比较/PWM 引脚 → Port B [10:8]
3. TMR3 保护 → Port B [7:6]
4. TMR3 比较输出 → Port B [5:0]

表 9-4 给出了控制信号和特殊功能引脚的描述。

表 9-4 端口 B 的特殊功能

	SFR Pin	Type	PHB	PL	OEB	描述
IOB[15:14]	-	-	-	-	-	-
IOB13	SDO/RXD1	O	1	0	SPISDOEB/ UARTX1OEB	SPIEN=1：输出为主从模式 UARTEN=1： 通用串口数据发送 UARTX1OEB =1：UART 传输数据输出
IOB12	SDI/TXD1	I	1	0	SPIEN / UARTRX1EN	当 SPIEN=1 时数据输入 / 当 UARTRX1EN =1 时 UART 接收数据输入
IOB11	SCK	I/O	1	0	SCKOEB	SPIEN=1，输入为从模式而输出为主模式
IOB10	TIO0A	User	1	0	TIO0AEN	P_TMR0_TGRA 输入捕获/PWM 输出引脚和位置侦测输入引脚
IOB9	TIO0B	User	1	0	TIO0BEN	P_TMR0_TGRB 输入捕获/PWM 输出引脚和位置侦测输入引脚
IOB8	TIO0C	User	1	0	TIO0CEN	P_TMR0_TGRC 输入捕获/PWM 输出引脚和位置侦测输入引脚
IOB7	OL1	I	1	0	OL1EN	过载保护输入 1
IOB6	FTINP1	I	1	0	FTIN1EN	外部故障保护输入 1

	SFR Pin	Type	PHB	PL	OEB	描述
IOB5	TIO3A	0	1	0	$\overline{\text{TIO3A/U1EN}}$ | TIO3HZ	U1 相输出引脚
IOB4	TIO3B	0	1	0	$\overline{\text{TIO3B/V1EN}}$ | TIO3HZ	V1 相输出引脚
IOB3	TIO3C	0	1	0	$\overline{\text{TIO3C/W1EN}}$ | TIO3HZ	W1 相输出引脚
IOB2	TIO3D	0	1	0	$\overline{\text{TIO3D/U1NEN}}$ | TIO3HZ	U1N 相输出引脚
IOB1	TIO3E	0	1	0	$\overline{\text{TIO3E/V1NEN}}$ | TIO3HZ	V1N 相输出引脚
IOB0	TIO3F	0	1	0	$\overline{\text{TIO3F/W1NEN}}$ | TIO3HZ	W1N 相输出引脚

9.5 IO 端口 C

端口 C 是一个多功能复用 IO 端口。端口 C 除了拥有普通的 I/O 功能外，还具有 PWM 输出、捕获输入功能和串行通信接口的功能。端口 C 的所功能都通过 5 个控制寄存器切换控制。

P_IOC_Data ($7070H)：IO 端口 C 的数据寄存器

向 P_IOC_Data 写入数据是写入 IO 的输出缓冲寄存器，而读取则是读管脚状态。

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Data							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Data							

P_IOC_Buffer ($7071H)：IO 端口 C 的缓冲寄存器

P_IOC_Buffer 写入与 P_IOC_Data 写入相同。而读取则是读 IO 的输出缓冲寄存器，利用这一特性，可实现单个 I/O 口的独立控制。

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Buffer							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Buffer							

注：读 P_IOC_Data 和读 P_IOC_Buffer 从不同的物理路径读取数据。读 P_IOC_Data 时的内容来自 I/O 端口 C 的引脚电平,而读 P_IOC_Buffer 内容则来自 I/O 端口 C 的输出缓冲寄存器。

P_IOC_Dir ($7072H)：IO 端口 C 的方向寄存器

端口 C 各 I/O 的方向设定寄存器。当相应的位为“1”时，I/O 被设为输出状态；当相应的位为“0”时，I/O 被设为输入状态。

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Dir							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Dir							

P_IOC_Attrib ($7073H)： IO 端口 C 属性寄存器

端口 C 各 I/O 的属性设定寄存器。它同 P_IOC_Dir 和 P_IOC_Data 相互配合可以将 I/O 设为多种属性。详情请见表 9-1

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
P_IOC_Attrib							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
P_IOC_Attrib							

P_IOC_SPE ($7082H)： IO 端口 C 的特殊功能寄存器

端口 B 的特殊功能使能寄存器。用来使能 IOC[15..2]的特殊功能。相应的位功能定义如下所述。

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	0	0
W2NEN	V2NEN	U2NEN	W2EN	V2EN	U2EN	FTIN2EN	OL2EN

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TIO1CEN	TIO1BEN	TIO1AEN	保留	EXINT1EN	EXINT0EN	保留	

第 15 位 **W2NEN：** W2N 引脚模式选择

0 = GPIO

1 = W2N 相

第 14 位 **V2NEN：** V2N 引脚模式选择

0 =GPIO

1 = V2N 相

第 13 位 **U2NEN：** U2N 引脚模式选择

0 =GPIO

1 = U2N 相

第 12 位 **W2EN：** W2 引脚模式选择

0 =GPIO

1 = W2 相

第 11 位 **V2EN：** V2 引脚模式选择

0 =GPIO

1 = V2 相

第 10 位 **U2EN：** U2 引脚模式选择

0 =GPIO

1 = U2 相

第 9 位 **FTIN2EN：** 外部故障保护输入 2 使能

0 = 禁止

1 = 使能

第 8 位 **OL2EN：** 过载保护输入 2 使能

0 = 禁止

1 = 使能

第 7 位 **TIO1CEN：** P_TMR1_TGRC 输入捕获/PWM 输出引脚和位置侦测输入使能

0 = 禁止

1 = 使能

第 6 位 **TIO1BEN：** P_TMR1_TGRB 输入捕获/PWM 输出引脚和位置侦测输入使能

0 = 禁止

1 = 使能

第 5 位 **TIO1AEN：** P_TMR1_TGRA 输入捕获/PWM 输出引脚和位置侦测输入使能

0 = 禁止

1 = 使能

第 4 位 保留

第 3 位 **EXINT1EN**： 外部中断输入 1 使能

0 = 禁止

1 = 使能

第 2 位 **EXINT0EN**： 外部中断输入 0 使能

0 = 禁止

1 = 使能

第 1:0 位 保留

注 : P_IOC_SPE[15 :10]默认 '1'

端口 C 的特殊功能

端口 C 与一下特殊功能共享：

1. TMR4 比较输出 → Port C [15:10]
2. TMR4 保护 → Port C [9:8]
3. TMR1 捕获/比较/PWM 引脚 → Port C [7:5]
4. 蜂鸣器输出 → Port C [4]
5. 外部输入引脚 → Port C [3:2]
6. UART 引脚 → Port C [1:0]

表 9-5 给出了控制信号与特殊功能引脚的描述

表 9-5 端口 C 的特殊功能表

	SFR 引脚	类型	PHB	PL	OEB	描述
IOC15	TIO4F/W2N	O	1	0	$\overline{\text{TIO4F/W2NEN}}$ | TIO4HZ	W2N 相输出引脚
IOC14	TIO4E/V2N	O	1	0	$\overline{\text{TIO4E/V2NEN}}$ | TIO4HZ	V2N 相输出引脚
IOC13	TIO4D/U2N	O	1	0	$\overline{\text{TIO4D/U2NEN}}$ | TIO4HZ	U2N 相输出引脚
IOC12	TIO4C/W2	O	1	0	$\overline{\text{TIO4C/W2EN}}$ | TIO4HZ	W2 相输出引脚
IOC11	TIO4B/V2	O	1	0	$\overline{\text{TIO4B/V2EN}}$ | TIO4HZ	V2 相输出引脚
IOC10	TIO4A/U2	O	1	0	$\overline{\text{TIO4A/U2EN}}$ | TIO4HZ	U2 相输出引脚
IOC9	FTIN2	I	1	0	FTIN2EN	外部故障保护输入引脚 2
IOC8	OL2	I	1	0	OL2EN	过载输入引脚 2
IOC7	TIO1C	User	1	0	TIO1CEN	TGRC1 捕获输入/PWM 输出引脚或位置侦测输入引脚

	SFR 引脚	类型	PHB	PL	OEB	描述
IOC6	TIO1B	User	1	0	TIO1BEN	TGRB1 输入捕获/PWM 输出引脚或位置侦测输入引脚
IOC5	TIO1A	User	1	0	TIO1AEN	TGRA1 输入捕获/PWM 输出引脚或位置侦测输入引脚
IOC4	BZO	O	1	0	BZOEB	蜂鸣器输出
IOC3	EXINT1	I	User	User	EXTINT1EN	外部中断输入 1
IOC2	EXINT0	I	User	User	EXTINT0EN	外部中断输入 0
IOC1	TXD2	O	1	0	UARTX2OEB	当 UARTEN=1 时，UART 传输数据输出；当 UARTX2OEB =1 时，Hi-Z
IOC0	RXD2	I	1	0	UARTRX2EN	当 UARTRX2EN =1 时，UART 接收数据输入

9.6 IO 端口 D

端口 D 是一个普通 IO 端口，只有普通 IO 的功能。但在 SPMC75F2313A 中，IOD[1..0]在 ICE（仿真调试接口）使能的情况下会用作仿真调试接口。

P_IOD_Data ($7078H)：IO 端口 D 的数据寄存器

向 P_IOD_Data 写入数据是写入 IO 的输出缓冲寄存器，而读取则是读管脚状态。

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Data							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Data							

P_IOD_Buffer ($7079H) IO 端口 D 的缓冲寄存器

P_IOD_Buffer 写入与 P_IOD_Data 写入相同。而读取则是读 IO 的输出缓冲寄存器，利用这一特性，可实现单个 I/O 口的独立控制。

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

P_IOD_Buffer

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Buffer							

注：读 P_IOD_Data 和读 P_IOD_Buffer 从不同的物理路径读取数据。读 P_IOD_Data 时的内容来自 I/O 端口 D 的引脚电平,而读 P_IOD_Buffer 内容则来自 I/O 端口 D 的输出缓冲寄存器。

P_IOD_Dir ($707AH)：IO 端口 D 的方向寄存器

端口 D 各 I/O 的方向设定寄存器。当相应的位为“1”时，I/O 被设为输出状态；当相应的位为“0”时，I/O 被设为输入状态。

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Dir							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Dir							

P_IOD_Attrib ($707BH)：IO 端口 D 的属性寄存器

端口 D 各 I/O 的属性设定寄存器。它同 P_IOC_Dir 和 P_IOC_Data 相互配合可以将 I/O 设为多种属性。详情请见表 9-1

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
P_IOD_Attrib							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
P_IOD_Attrib							

端口 D 的特殊功能

端口 D 与下面的特殊功能如表 9-6 所示：

表 9-6 端口 D 的特殊功能

	SFR Pin	Type	PHB	PL	OEB	描述
IOD[15:2]	–	–	–	–	–	–
IOD1	ICESDA	I/O	1	0	ICESDAOEB	**ICE 串行地址或数据输入/输出**
IOD0	ICECLK	I	1	0	ICECLKEN	ICE 串行时钟输入

9.7 I/O 端口编程注意事项

当使用这 GPIO 端口时，为确保预期功能的实现，必须认真考虑每个属性的设置。

【例 9-1】设 IOA[3:0]为下拉，IOA[7:4]为上拉，IOA[11:8]为低电平输出，IOA[15:12]为高电平输出

```
P_IOA_SPE->W          =    0x0000;          /* 禁止特殊功能 */
P_IOA_Dir->W          =    0xFF00;
P_IOA_Attrib->W       =    0xFF00;
P_IOA_Buffer->W       =    0xF0F0;
P_IOA_Buffer->B.Bit14 =    1;
```

例 9-1 IO 编程设计参考 1

【例 9-2】从 IOA 中读取数据有两种的方式：读取 P_IOA_Data 的读取的是在 I/O 引脚的状态；读取 P_IOA_Buffer 是读取在缓冲寄存器上的数据。

```
Data    = P_IOA_Data->W;
Data    = P_IOA_Buffer->W;
```

例 9-2 IO 编程设计参考 2

【例 9-3】共有两种向 IOA 写入数据的方式。向 P_IOA_Data 写入和向 P_IOA_Buffer 写入。

```
P_IOA_SPE->W          =    0x0000;          /*禁止特殊功能*/
    P_IOA_Data->W     =    0xFFFF;
P_IOA_Data->B.Bit1    =    0;
P_IOA_Buffer->W       =    0x5555;
P_IOA_Buffer->B.Bit0  =    0;
```

例 9-3 IO 编程设计参考 3

9.8 I/O 初始化

当发生上电复位、低电压复位、外部复位及 ICE 复位等状态时，IO 端口将重新初始化 ，如表 9-7 所示。然而看门狗复位和非法地址，非法指令复位时，IO 端口会保持初始状态。

表 9-7 复位后的状态

端口名称	寄存器名称	地址	初始值
Port A 端口 A	P_IOA_Data	$7060H	0000,0000,0000,0000B
	P_IOA_Buffer	$7061H	0000,0000,0000,0000B

端口名称	寄存器名称	地址	初始值
	P_IOA_Dir	$7062H	0000,0000,0000,0000B
	P_IOA_Attrib	$7063H	0000,0000,0000,0000B
	P_IOA_Latch	$7064H	0000,0000,0000,0000B
	P_IOA_SFE	$7080H	0000,0000,0000,0000B
	P_IOA_KCER	$7084H	0000,0000,0000,0000B
Port B 端口 B	P_IOB_Data	$7068H	0000,0000,0000,0000B
	P_IOB_Buffer	$7069H	0000,0000,0000,0000B
	P_IOB_Dir	$706AH	0000,0000,0000,0000B
	P_IOB_Attrib	$706BH	0000,0000,0000,0000B
	P_IOB_SFE	$7082H	0000,0000,0011,1111B
Port C 端口 C	P_IOC_Data	$7070H	0000,0000,0000,0000B
	P_IOC_Buffer	$7071H	0000,0000,0000,0000B
	P_IOC_Dir	$7072H	0000,0000,0000,0000B
	P_IOC_Attrib	$7073H	0000,0000,0000,0000B
	P_IOC_SFE	$7083H	1111,1100,0000,0000B
Port D 端口 D	P_IOD_Data	$7078H	0000,0000,0000,0000B
	P_IOD_Buffer	$7079H	0000,0000,0000,0000B
	P_IOD_Dir	$707AH	0000,0000,0000,0000B
	P_IOD_Attrib	$707BH	0000,0000,0000,0000B

第10章 节电模式和唤醒功能

10.1简介

SPMC75X 系列微控制器有标准模式和两种节电模式(Wait 和 Standby)相应功能如下：

- **标准模式**

  芯片在标准模式下运行时所有内部模块可用，芯片的耗电最大。

- **Wait**

  Wait 模式下，只有 CPU 掉电停止工作以降低功耗。其他外设保持着先前的状态并且功能可用。一旦唤醒，CPU 将继续工作，执行接下的指令。图 10-1 描述 Wait 模式时序

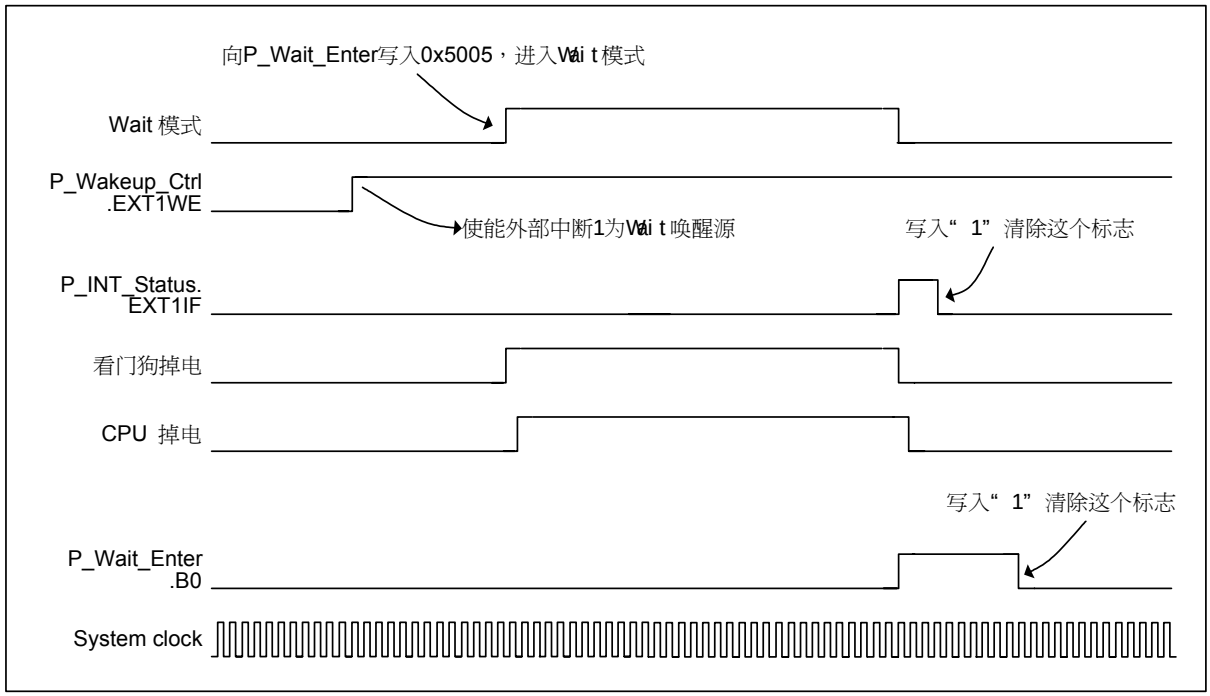


图 10-1 Wait 模式时序

- **Standby**

  Standby 模式下所有的模块都变为无效，此时功耗达到最小。唤醒后，CPU 复位并回到标准运行模式。注意，如果 MCP 定时器 3 或定时器 4 已经处于 PWM 输出模式下时，芯片不会进入 Wait 或 Standby。图 10-2 描述 Standby 时序

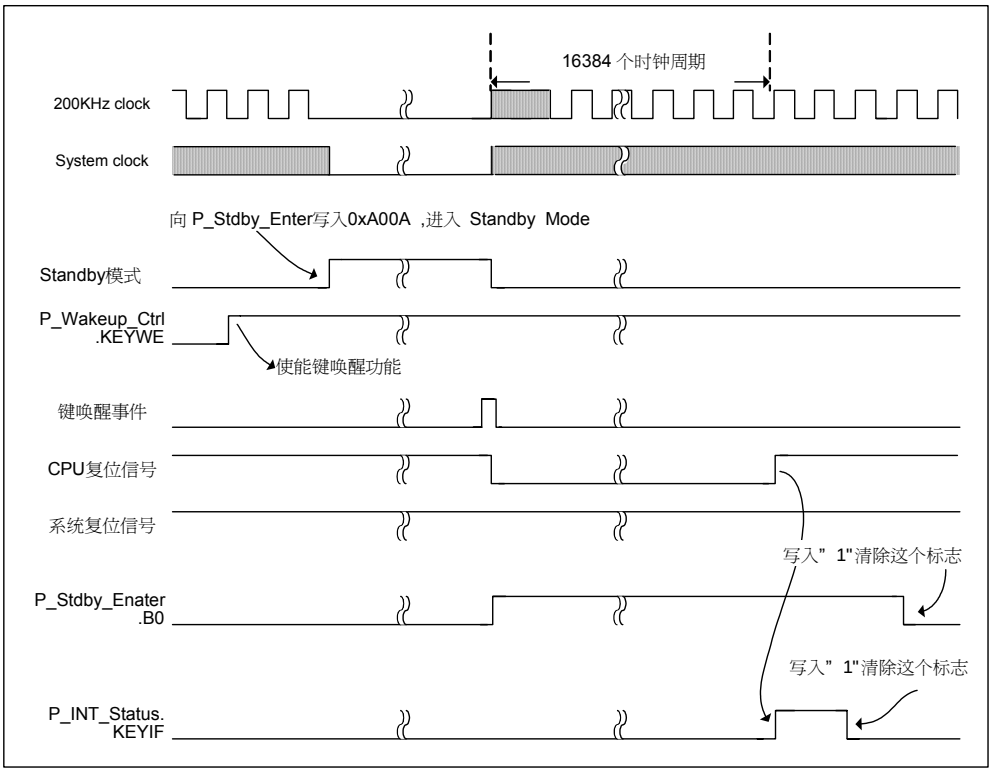


图 10-2 Standby 模式时序

Standby 模式下所有功能都会关闭。如果按键唤醒功能为有效，当键唤醒功能使能，Wait 和 Standby 这两种模式都可以通过按键唤醒,图 10-3 描述标准模式和两种节电模式(Wait 和 Standby)之间的转换。

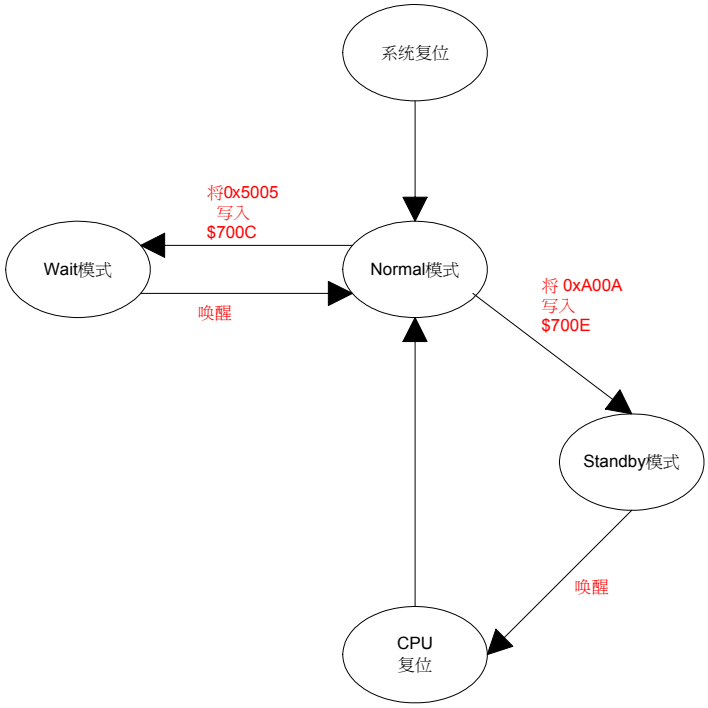


图 10-3 标准模式和两种节电模式(Wait 模式和 Standby 模式)之间的转换

表 10-1 节电模式和各模块运行状态的关系

	Wait	Standby
CPU	关闭	关闭
PLL	工作	关闭
唤醒后	执行下一条指令	CPU 复位

10.2唤醒源

Standby 模式只能由键唤醒源唤醒，而 Wait 模式的唤醒事件有以下 28 个来源：

- **定时器/PWM 模块：**
  -定时器 0：TPR_0，TGRA_0，TGRB_0，TGRC_0,位置改变侦测、上溢、下溢
  -定时器 1：TPR_1，TGRA_1，TGRB_1，TGRC_1,位置改变侦测、上溢、下溢
  -定时器 2：TPR_2，TGRA_2，TGRB_2
  -定时器 3：TPR_3，TGRD_3
  -定时器 4：TPR_4，TGRD_4
- **比较匹配定时器：**CMT_0 比较匹配中断、CMT_1 比较匹配中断
- **键唤醒**
- **外部中断：**EXINT0、EXINT1
- **串行通讯接口：**UART 发送中断、UART 接收中断、SPI 发送中断、SPI 接收中断

10.3控制寄存器

SPMC75X 系列微控制器的低功耗控制模块共有 3 个控制寄存器，如表 10-2 所示。通过这 3 个控制寄存器可以完成低功耗控制模块所有功能的控制。

表 10-2 低功耗控制模块的控制寄存器

地址	寄存器	名称
700Ch	P_Wait_Enter	Wait模式入口寄存器
700Eh	P_Stdby_Enter	Standby模式入口寄存器
700Fh	P_Wakeup_Ctrl	唤醒控制寄存器

P_Wait_Enter ($700C)：Wait 模式入口寄存器

B15	B14	B13	B12	B11	B10	B9	B8
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
WaitCMD							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	R/W
0	0	0	0	0	0	0	0
WaitCMD							

1. 写入”5005h”进入 Wait 模式(CPU 关闭，锁相环电路工作)。
2. Bit0 为 Wait 唤醒标志，如果读出 1 说明芯片从 Wait 模式唤醒。
3. 写入”0001h”清除唤醒 Wait 标志。
4. 注意 在进入 Wait 模式之前， MCP 定时器 3 或者 4 不能设置为 PWM 输出状态.仿真时不能进 Wait 模式.

P_Stdby_Enter ($700E)：Standby 模式入口寄存器

B15	B14	B13	B12	B11	B10	B9	B8
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
StdbyCMD							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	R/W
0	0	0	0	0	0	0	0
StdbyCMD							

1. 写入”A00Ah”,进入 Standby 模式(CPU 关闭，锁相环电路工作)。
2. Bit0 为 Standby 唤醒标志，如果读出 1 说明芯片从 Standby 模式唤醒。
3. 写入”0001h”清除 Standby 唤醒标志。
4. 在 MCP 定时器设置为 PWM 输出状态或是仿真时均不能进入 Standby 模式。

P_Wakeup_Ctrl ($700F)：唤醒控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
KEYWE	UARTWE	SPIWE	EXT1WE	EXT0WE	保留		

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TPM2WE	PDC1WE	PDC0WE	CMTWE	保留			

第 15 位 **KEYWE**：键唤醒使能位

0 ＝ 禁止

1 ＝ 使能

第 14 位 **UARTWE**：UART 端口唤醒使能位

0 = 禁止

1 = 使能

第 13 位 **SPIWE**：SPI 端口唤醒使能位

0 = 禁止

1 = 使能

第 12 位 **EXT1WE**：外部中断 1 唤醒使能位

0 = 禁止

1 = 使能

第 11 位 **EXT0WE**：外部中断 0 唤醒使能位

0 = 禁止

1 = 使能

第 10:8 位 保留

第 7 位 **TPM2WE**：TPM 定时器 2 唤醒使能位

0 = 禁止

1 = 使能

第 6 位 **PDC1WE**：PDC 定时器 1 唤醒使能位

0 = 禁止

1 = 使能

第 5 位 **PDC0WE**：PDC 定时器 0 唤醒使能位

0 = 禁止

1 = 使能

第 4 位 **CMTWE**：比较匹配定时器唤醒使能位

0 = 禁止

1 = 使能

第 3:0 位 保留

10.4设计参考

【例 10-1】将 MCU 设置为 Wait 模式，清除 Wait 模式标志

```
P_Wait_Enter->W =  CW_WaitCMD;                    /* 进入 Wait 模式 */
P_Wait_Enter->W =  CW_WaitClr;                    /* 清除 Wait 模式标志 */
```

例 10-1 MCUWait 模式设计参考

【例 10-2】将 MCU 设置为 Standby 模式，清除 Standby 模式标志

```
P_Stdby_Enter->W = CW_StdbyCMD;                   /* 进入 Standby 模式 */
P_Stdby_Enter->W = CW_StdbyClr;                   /* 清除 Standby 模式标志 */
```

例 10-2 MCU Standby 模式设计参考

第11章 PDC 定时器模块

11.1简介

SPMC75X 系列微控制器提供了两个 PDC(Phase Detection Control)定时器，PDC 定时器 0 和 PDC 定时器 1，用于捕获功能和产生 PWM 波形输出，同时具有侦测无刷直流电机位置改变的特性。PDC 定时器非常适用于机械速度的计算，其包括交流感应电机和无刷直流电机，侦测无刷直流电机(转子)位置而控制其换流。PDC 定时器 0 和定时器 1 的整体框图如图 11-1 所示。PDC 定时器的详细规格说明见表 11-1。

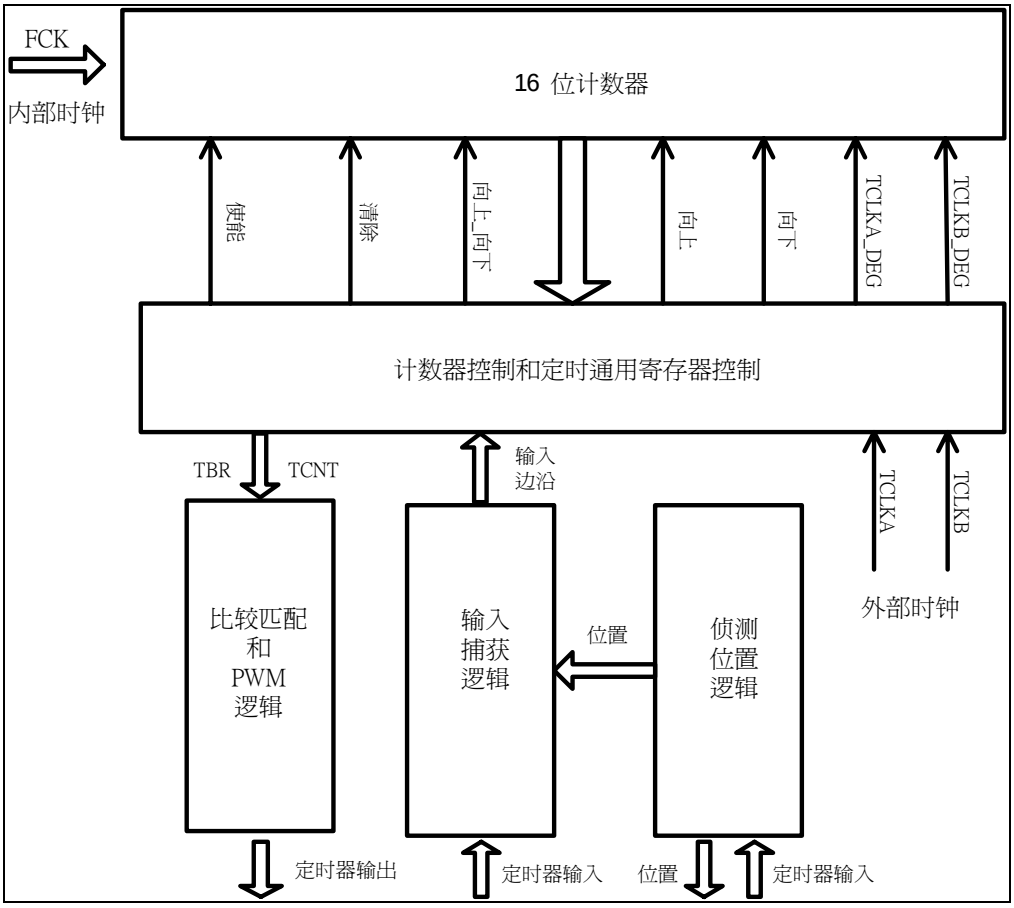


图 11-1 PDC 定时器功能示意框图

11.2PDC 定时器的功能

- 能够处理总计6路(每个定时器3路)捕获或霍尔信号输入。或者输出6路PWM波形(每个定时器3路)。
- 6个定时通用寄存器(TGRAx/TGRBx/TGRCx, x = 0, 1)：每个通道各有3个寄存器作为PWM输出或输入捕获功能。
- 6个定时缓冲寄存器(TBRAx/TBRBx/TBRCx, x = 0, 1)：每个通道各有3个缓冲寄存器作为

PWM缓冲输出或输入捕获功能。

- ❑ 8个可编程的时钟源：6个内部时钟(F_{CK}/1, F_{CK}/4, F_{CK}/16, F_{CK}/64, F_{CK}/256, F_{CK}/1024)，两个外部时钟(TCLKA 和TCLKB)。
- ❑ 可编程的定时器操作模式
  1. 定时计数模式
     - 标准计数模式：连续递增计数
     - PWM功能：在比较匹配和输出保持时既可以选择输出0也可以选择输出1。
     - 输入捕获功能:选择由上升沿、下降沿、双沿(上升、下降)触发捕获或位置改变侦 测。
  2. PWM模式
     - – 每个定时器能输出3个独立的占空比可设定的PWM。
  3. 边沿PWM波形发生模式
     - – PWM以标准递增计数方式输出
  4. 中心(Center-aligned)PWM波形发生模式
     - – PWM以递增/递减模式输出
  5. 侦测位置改变模块(PDC)
     - 可编程的位置信号输入采样时钟 ：4个内部时钟源(F_{CK}/4, F_{CK}/8, F_{CK}/32 和F_{CK}/128)
     - 可编程的位置采样模式：PWM有输出时采样、周期性采样、下相(UN, VN, WN)导通时采样
     - 可编程的位置信号采样计数，以避免脉冲干扰位置数据。
- ❑ 总计14个中断源
  - 定时器周期比较匹配中断和计数器上溢/下溢中断源
  - 3个TGR寄存器比较匹配中断源
  - 侦测位置改变中断源
- ❑ 定时器缓冲操作
  - 输入捕获寄存器可以构成双缓冲器。PWM 占空比寄存器 (TGRA/B/C)会在周期中断时同步更新。
- ❑ 定时器的比较匹配和周期初始值可以任意设定
- ❑ 只读的16位递增/递减计数寄存器P_TMRx_TCNT (x = 0, 1)。
- ❑ 16位可读写的周期寄存器：P_TMRx_TPR (x = 0, 1)，提供系统时间基准。
- ❑ 3个16位的定时通用寄存器：P_TMRx_TGRA, P_TMRx_TGRB, P_TMRx_TGRC (x = 0, 1)，用于输入捕获和PWM输出。
- ❑ 3个只读定时缓冲寄存器：P_TMRx_TBRA, P_TMRx_TBRB, P_TMRx_TBRC (x = 0, 1)，是以上提到的定时通用寄存器的缓冲器。
- ❑ 16位可读写定时控制寄存器P_TMRx_Ctrl和输入输出控制寄存器P_TMRx_IOCtrl (x = 0, 1)，后者是进行捕获输入或者PWM输出模式的选择。
- ❑ 16位可读写定时中断使能寄存器P_TMRx_INT，提供7个中断源，和一个16位可读可写中断状态寄存器P_TMRx_Status (x = 0, 1)，记录中断的标志。

- 16 位可读写位置侦测控制寄存器 P_POSx_DectCtrl 和位置侦测数据寄存器 P_POSx_DectData (x = 0, 1) 用于位置信号反馈。

表 11-1 PDC 定时器规格

功能		PDC 定时器 0	PDC 定时器 1
时钟源		内部时钟: $F_{CK}/1$, $F_{CK}/4$, $F_{CK}/16$, $F_{CK}/64$ $F_{CK}/256$, $F_{CK}/1024$ 外部时钟: TCLKA, TCLKB	内部时钟: $F_{CK}/1$, $F_{CK}/4$, $F_{CK}/16$, $F_{CK}/64$ $F_{CK}/256$, $F_{CK}/1024$ 外部时钟: TCLKA, TCLKB
IO 引脚		TIO0A, TIO0B, TIO0C	TIO1A, TIO1B, TIO1C
定时通用寄存器		P_TMR0_TGRA, P_TMR0_TGRB, P_TMR0_TGRC	P_TMR1_TGRA, P_TMR1_TGRB, P_TMR1_TGRC
定时缓冲寄存器		P_TMR0_TBRA, P_TMR0_TBRB, P_TMR0_TBRC	P_TMR1_TBRA, P_TMR1_TBRB, P_TMR1_TBRC
定时周期和计数寄存器		P_TMR0_TPR, P_TMR0_TCNT	P_TMR1_TPR, P_TMR1_TCNT
捕获采样时钟		内部时钟: $F_{CK}/1$, $F_{CK}/2$, $F_{CK}/4$, $F_{CK}/8$	内部时钟: $F_{CK}/1$, $F_{CK}/2$, $F_{CK}/4$, $F_{CK}/8$
计数边沿		上升，下降，双沿计数	上升，下降，双沿计数
计数清除源		根据P_TMR0_TGRA, P_TMR0_TGRB, P_TMR0_TGRC捕获输入清除 根据P_POS0_DectData侦测位置改变数据变化清除 根据P_TMR0_TPR比较匹配清除	根据P_TMR1_TGRA, P_TMR1_TGRB, P_TMR1_TGRC捕获输入清除 根据P_POS1_DectData侦测位置改变数据变化清除 根据P_TMR1_TPR比较匹配清除
输入捕获功能		Yes	Yes
PWM 比较匹配输出功能	1 输出	Yes	Yes
	0 输出	Yes	Yes
	输出保持	Yes	Yes
边沿 PWM		Yes	Yes
中心PWM		Yes	Yes
相位计数模式		Yes, 相位输入为TCLKA/TCLKB	Yes, 相位输入为TCLKC/TCLKD
定期器缓冲操作		Yes	Yes
AD 转换触发		P_TMR0_TGRA 比较匹配	P_TMR1_TGRA比较匹配
中断源		定时器0 TPR 中断 定时器 0 TGRA 中断 定时器 0 TGRB 中断 定时器 0 TGRC 中断 定时器 0 PDC 中断 定时器 0 上溢 中断 定时器 0 下溢 中断	定时器 1 TPR 中断 定时器 1 TGRA 中断 定时器 1 TGRB 中断 定时器 1 TGRC 中断 定时器 1 PDC 中断 定时器 1 上溢中断 定时器1 下溢 中断

11.3PDC 定时器输入/输出/特殊功能管脚

通道	管脚名称	I/O	功能
PDC 定时器外部时钟输入	TCLKA	输入	外部时钟 A 输入(定时器 0 相位计数模式 A 相输入)
	TCLKB	输入	外部时钟 B 输入(定时器 0 相位计数模式 B 相输入)
	TCLKC	输入	外部时钟 C 输入(定时器 1 相位计数模式 C 相输入)
	TCLKD	输入	外部时钟 D 输入(定时器 1 相位计数模式 D 相输入)
PDC 定时器 0	TIO0A	I/O	P_TMR0_TGRA 输入捕获/PWM 输出引脚
	TIO0B	I/O	P_TMR0_TGRB 输入捕获/PWM 输出引脚
	TIO0C	I/O	P_TMR0_TGRC 输入捕获/PWM 输出引脚
PDC 定时器 1	TIO1A	I/O	P_TMR1_TGRA 输入捕获/PWM 输出引脚
	TIO1B	I/O	P_TMR1_TGRB 输入捕获/PWM 输出引脚
	TIO1C	I/O	P_TMR1_TGRC 输入捕获/PWM 输出引脚

11.4PDC 定时器的工作模式

SPMC75X 系列微控制器内 PDC 定时器具有以下 5 种工作模式，P_TMRx_Ctrl (x = 0, 1)寄存器中的 MODE 的值决定了 PDC 定时计数的模式。当定时器启动寄存器 P_TMR_Start 相应位置 1 时，定时器就从 0x0000 开始计数。并且 P_TMRx_Ctrl (x = 0, 1)寄存器中的 CCLS 位的值决定了计数清除源。

- 标准操作(普通的递增计数)
- 相位计数模式 1-4
- 依靠外部时钟输入引脚 TCLKA 或 TCLKB 计数
- 边沿 PWM 模式(连续计数，PWM 输出模式)
- 中心 PWM 模式(递增/递减计数，PWM 输出模式)

11.4.1 连续递增计数和边沿 PWM 输出模式

每个 PDC 定时器都可由 P_TMRx_Ctrl (x = 0, 1)寄存器的 MODE 位设置为边沿 PWM 输出模式或没有任何 PWM 输出的标准操作模式。计数器从 0x0000 开始以递增方式计数，一直计数到与定时周期寄存器的设定值匹配为止，在这种模式下，用户需要设定 P_TMRx_TPR (x = 0，1)寄存器，并将计数清除源(CCLS)设置成由周期比较匹配来清除。

PDC 定时器进行连续递增计数，其输入时钟源依据相应定时控制寄存器中 TMRPS 的值所决定。当定时器计数的值与定时器周期寄存器的值相等时，周期比较匹配中断标志 TPRIF 置位，定时计数器清零。如果 P_TMRx_INT (x = 0, 1)寄存器中的 PPRIE 位使能，则产生周期中断请求；当 P_TMRx_INT (x = 0, 1)寄存器中的下溢中断使能 TCUIE 置位时，当计数寄存器变化到 0x0000 时，将产生计数下溢中断请求；当 P_TMRx_INT (x = 0, 1)寄存器中的上溢中断使能 TCVIE 置位时，当计数寄存器递增到 0xFFFF 时，将产生计数上溢中断请求；当定时计数器的值与 TGRA、TGRB 或 TGRC 匹配上后，就发生了通用寄存器比较匹配事件，如果定时中断控制寄存器的相应位 TGAIE、TGBIE 或 TGCIE 为使能状态，则产生通用寄存器比较匹配中断。

P_TMRx_TPR (x = 0, 1)的初始值可以是 0x0000 到 0xFFFF 之间的任何数值。定时器的时钟源既可以选择外部时钟输入也可以选择内部时钟源。PDC 定时器的标准递增计数模式如图 11-2 所示。

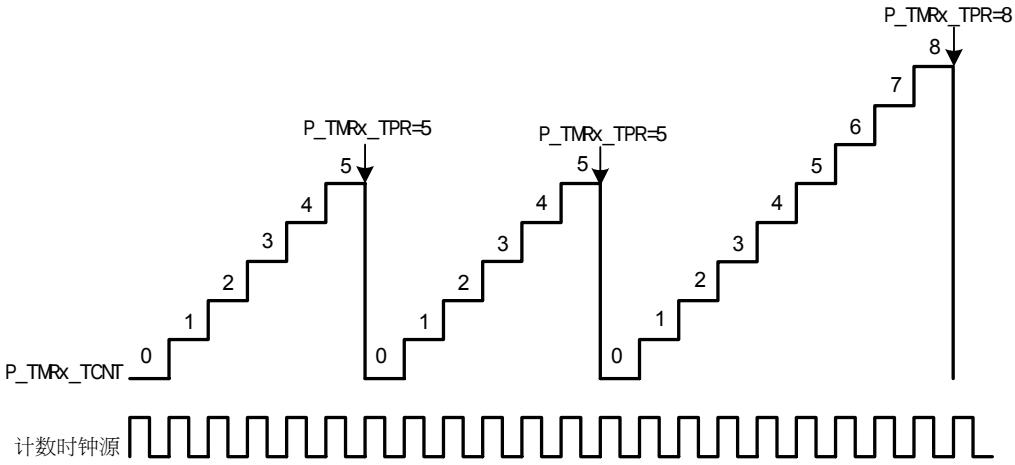


图 11-2 连续递增计数模式(CCLS = 111'b, CKEGS = 000'b, TMRPS = 000'b)

在边沿 PWM 模式下，用户必须设置 P_TMRx_TPR (x= 0, 1)周期寄存器和 P_TMRx_TGRy (y = A, B, C)通用寄存器，然后将计数清除源(CCLS)设置为定时器周期比较匹配事件清除。比较匹配输出状态设置是在 P_TMRx_IOCtrl (x= 0, 1)控制寄存器中。PDC 定时器 0 的边沿 PWM 标准连续递增计数模式如图 11-3 所示,图 11-4 描述边沿 PWM 输出模式的时序。

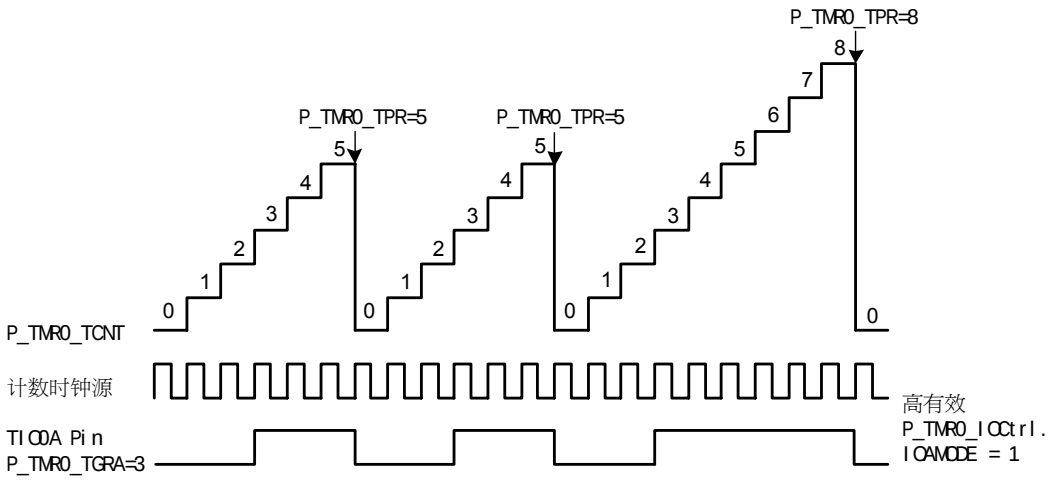


图 11-3 边沿 PWM 模式

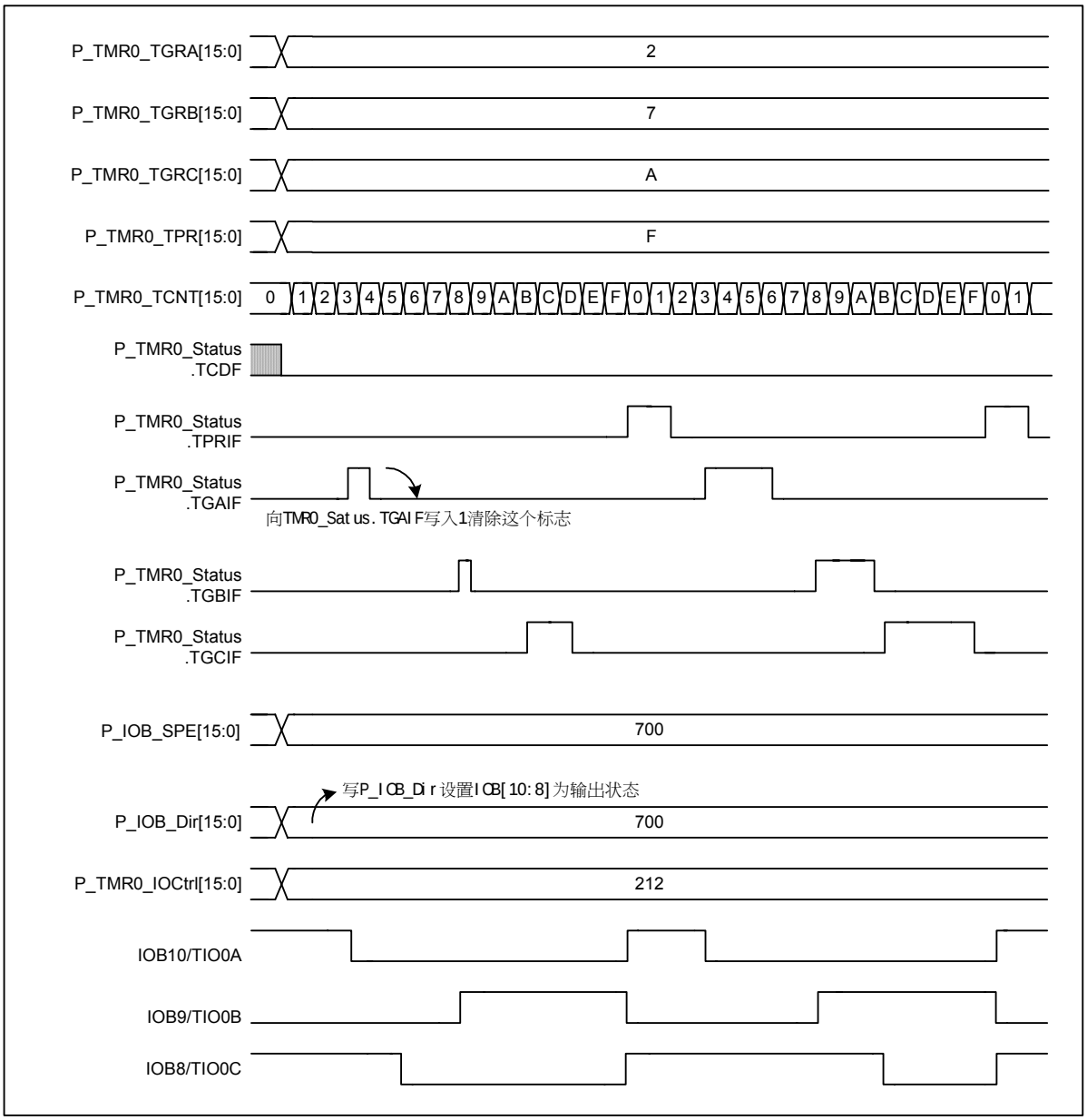


图 11-4 边沿 PWM 模式时序

11.4.2 连续递增/递减计数和中心 PWM 输出模式

连续递增/递减计数模式的操作与递增计数模式基本相同，唯一不同的是：定时器周期寄存器设置了整个计数过程的中间过渡点，计数过程中先递增，当达到定时周期寄存器设定值后开始递减。定时器的周期是寄存器 P_TMRx_TPR (x = 0, 1)中设置值的两倍。连续递增/递减计数模式如图 11-5 所示。

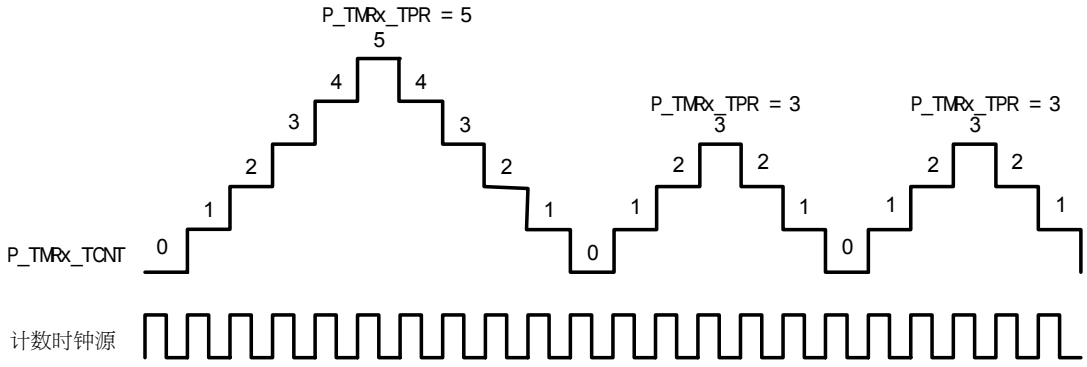


图 11-5 连续递增/递减计数模式(CCLS = 111'b, CKEGS = 00'b, TMRPS = 000'b)

PDC 定时器周期寄存器的初始值可以是 0x0000 到 0xFFFF 中的任何数值，当计数寄存器的值与周期定时寄存器的值相等的时候，PDC 定时器开始递减计数直到零。在这个过程中，上溢、下溢中断的发生情况与递增计数模式的描述相同。计数的方向由 P_TMRx_Status (x = 0, 1)寄存器的 TCDF 位显示。定时器的时钟源既可以选择外部时钟输入也可以选择内部的 F_{CK} 时钟源。如图 11-6 所示为 PDC 定时器 0 在递增/递减计数模式下的中心 PWM 模式。

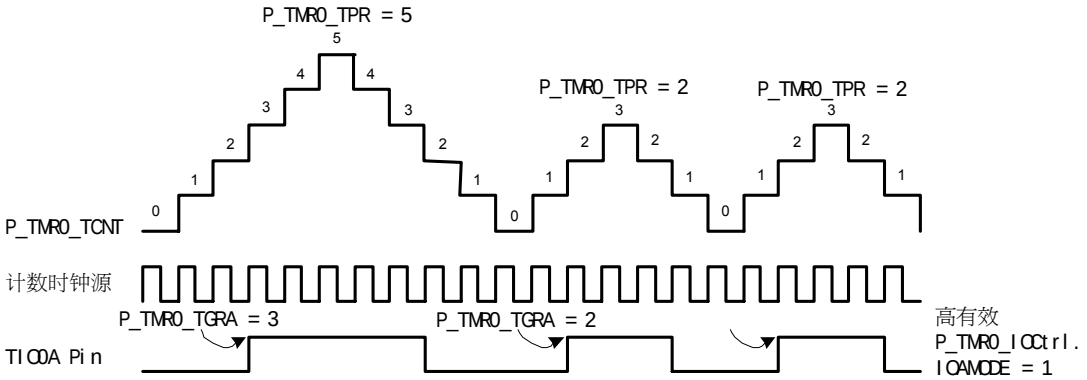


图 11-6 中心 PWM 模式

11.5PDC 定时器控制寄存器

SPMC75X 系列微控制器的 PDC 定时器 0 和 PDC 定时器 1 模块共有 29 个控制寄存器，如表 11-1 所示。通过这 29 个控制寄存器可以完成 PDC 定时器 0 和 PDC 定时器 1 模块所有功能的控制。

表 11-1 PDC 定时器 0 和 PDC 定时器 1 的控制寄存器

地址	寄存器	名称
7400h	P_TMR0_Ctrl	定时器0控制寄存器
7410h	P_TMR0_IOCtrl	定时器0输入/输出控制寄存器
7420h	P_TMR0_INT	定时器0中断使能寄存器
7425h	P_TMR0_Status	定时器0中断状态寄存器
7462h	P_POS0_DectCtrl	定时器0位置侦测控制寄存器
7464h	P_POS0_DectData	定时器0位置侦测数据寄存器

地址	寄存器	名称
7430h	P_TMR0_TCNT	定时器0计数寄存器
7435h	P_TMR0_TPR	定时器0周期寄存器
7440h	P_TMR0_TGRA	定时器0通用寄存器A
7441h	P_TMR0_TGRB	定时器0通用寄存器B
7442h	P_TMR0_TGRC	定时器0通用寄存器C
7450h	P_TMR0_TBRA	定时器0缓冲寄存器A
7451h	P_TMR0_TBRB	定时器0缓冲寄存器B
7452h	P_TMR0_TBRC	定时器 0 缓冲寄存器 C
7401h	P_TMR1_Ctrl	定时器1控制寄存器
7411h	P_TMR1_IOCtrl	定时器1输入/输出寄存器
7421h	P_TMR1_INT	定时器1中断使能寄存器
7436h	P_TMR1_Status	定时器1中断状态寄存器
7463h	P_POS1_DectCtrl	定时器1位置侦测控制寄存器
7465h	P_POS1_DectData	定时器1位置侦测数据寄存器
7431h	P_TMR1_TCNT	定时器1计数寄存器
7436h	P_TMR1_TPR	定时器1周期寄存器
7443h	P_TMR1_TGRA	定时器1通用寄存器A
7444h	P_TMR1_TGRB	定时器1通用寄存器B
7445h	P_TMR1_TGRC	定时器1通用寄存器C
7453h	P_TMR1_TBRA	定时器1缓冲寄存器A
7454h	P_TMR1_TBRB	定时器1缓冲寄存器B
7455h	P_TMR1_TBRC	定时器 1 缓冲寄存器 C
7405h	P_TMR_Start	定时器计数启动寄存器

11.5.1 PDC 定时器的控制寄存器

寄存器 P_TMRx_Ctrl (x = 0, 1)设置时钟源的选择、计数时钟边沿、计数器清除源、计数器清除边沿、输入捕获采样时钟和时钟操作模式。TCLKA, TCLKB 引脚的时钟输入将由系统时钟 F_{CK} 采样，任何小于四倍采样时钟脉冲宽度的信号都会被忽略。当设置为双沿触发计数时，输入时钟频率加倍。每个 PDC 定时器都可由 P_TMRx_Ctrl (x = 0, 1)寄存器的 MODE 位设置为 PWM 输出模式或没有 PWM 输出的标准操作模式。当 MODE 位设置为相位计数模式时，则定时器 0 的输入是 TCLKA/TCLKB 的信号，定时器 1 的输入是 TCLKC/TCLKD 的信号。相位计数模式下时钟源应为内部时钟。

P_TMR0_Ctrl ($7400)：PDC 定时器 0 控制寄存器

P_TMR1_Ctrl ($7401)：PDC 定时器 1 控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

SPCK	MODE	CLEGS

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CCLS			CKEGS		TMRPS		

第 15:14 位 **SPCK**：捕获输入采样时钟选择位，用于选择捕获输入的采样时钟，大于采样时钟 1/4 频率的输入将被忽略。

00 = FCK/1

01 = FCK/2

10 = FCK/4

11 = FCK/8

第 13:10 位 **MODE**：模式选择位。用于选择定时器操作模式。

0000 = 标准模式(连续递增计数)

0100 = 相位计数模式 1

0101 = 相位计数模式 2

0110 = 相位计数模式 3

0111 = 相位计数模式 4

1x0x = 边沿 PWM 模式(连续递增计数，PWM 输出)

1x1x = 中心 PWM 模式(连续递增/递减计数，PWM 输出)

第 9:8 位 **CLEGS**：计数器清除边沿选择位，用于输入捕获模式下选择计数器清除边沿。

00 =不清除

01 =上升沿

10 =下降沿

11 = 双沿

第 7:5 位 **CCLS**：计数器清除源选择位，用于选择 TCNT 的计数清除源。

000 = 禁止进行 TCNT 清除

001 = 由 P_TMRx_TGRA (x = 0, 1)清除 TCNT

010 = 由 P_TMRx_TGRB (x = 0, 1)清除 TCNT

011 = 由 P_TMRx_TGRC (x = 0, 1)清除 TCNT

100 = 每 6 次 P_POSx_DectData (x = 0, 1)变化清除 1 次 TCNT

101 = 每 3 次 P_POSx_DectData (x = 0, 1)变化清除 1 次 TCNT

110 = 每次 P_POSx_DectData (x = 0, 1)变化清除 1 次 TCNT

111 = P_TMRx_TPR (x = 0, 1)比较匹配事件发生，清除 1 次 TCNT

第 4:3 位 **CKEGS**：时钟边沿选择位，用于选择输入的时钟边沿触发方式。当输入时

钟为双沿计数时，输入时钟周期会减半。若选择 $F_{CK}/1$ 为计数时钟，如果为双沿触发，那么计数器将以上升沿方式工作。

00 = 上升沿计数

01 = 下降沿计数

1X = 双沿计数

第 2:0 位 **TMRPS**：定时器分频选择，用于选择 TCNT 计数时钟源。每个通道可分别的选择不同的时钟源。

000 = F_{CK} /1 下计数

001 = F_{CK} /4 下计数

010 = F_{CK} /16 下计数

011 = F_{CK} /64 下计数

100 = F_{CK} /256 下计数

101 = F_{CK} /1024 下计数

110 = 以 TCLKA 引脚输入时钟计数

111 =以 TCLKB 引脚输入时钟计数

11.5.2 PDC 定时器的输入输出控制寄存器

P_TMRx_IOCtrl (x =0, 1)寄存器控制着 TIOxA、TIOxB、TIOxC (x = 0, 1)引脚上的 PWM 输出或捕获输入操作方式。通过设置位于 P_TMRx_Ctrl (x = 0, 1)寄存器中的 CCLS 位和 MODE 位可以确定定时器输入输出动作的模式。当选择 PWM 输出模式时，IOAMODE/IOBMODE/IOCMODE 这些位决定着波形输出模式。当选择捕获输入模式时，IOAMODE/IOBMODE/IOCMODE 这些位则对包括侦测位置改变事件在内的捕获操作进行设置。

P_TMR0_IOCtrl ($7410) : PDC 定时器 0 输入输出控制寄存器

P_TMR1_IOCtrl ($7411) : PDC 定时器 1 输入输出控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
保留				IOCMODE			

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
IOBMODE				IOAMODE			

第 15:12 位 保留

第 11:8 位 **IOCMODE**：选择定时器 0/定时器 1 端口 C 设置

第 7:4 位 **IOBMODE**：选择定时器 0/定时器 1 端口 B 设置

第 3:0 位　　IOAMODE：选择定时器 0/定时器 1 端口 A 设置

PWM 比较匹配输出模式：

0000 ＝ 初始输出 0，当比较匹配时输出 0

0001 ＝ 初始输出 0，当比较匹配时输出 1

0010 ＝ 初始输出 1，当比较匹配时输出 0

0011 ＝ 初始输出 1，当比较匹配时输出 1

101xx ＝ 输出保持

输入捕获模式：

1000 ＝上升沿发生捕获事件中断

1001 ＝下降沿发生捕获事件中断

101x ＝ 双沿发生捕获事件中断

11xx ＝ 当侦测到位置改变时捕获输入(捕获 TCNT 寄存器值到 TGR 寄存器)并产生中断(仅适用于 TGRA 寄存器)

11.5.3　PDC 定时器的中断使能寄存器

P_TMRx_INT (x = 0, 1)寄存器用来设置以下功能：当 TGRA 比较匹配完成后是否启动 A/D 转换、发生侦测位置改变中断请求、TCNT 上溢/下溢、TGRA、TGRB、TGRC 周期寄存器比较匹配和输入捕获/比较匹配中断请求。

P_TMR0_INT ($7420)：PDC 定时器 0 中断使能寄存器

P_TMR1_INT ($7421)：PDC 定时器 1 中断使能寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R/W
0	0	0	0	0	0	0	0
保留							PDCIE

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TADSE	TCUIE	TCVIE	TPRIE	保留	TGCIE	TGBIE	TGAIE

第 15:9 位　保留

第 8 位　　PDCIE：侦测位置改变中断使能位。使能或禁止来自 P_POSx_DectData (x = 0, 1)寄存器的侦测位置改变中断请求。

0 ＝ 禁止

1 ＝ 使能

第 7 位　　TADSE：A/D 转换启动请求使能位。使能或禁止当 TGRA 比较匹配完成后是否需要　进行 A/D 转换。

0 = 禁止

1 = 使能

第 6 位 TCUIE：下溢中断使能位。使能或禁止由计数器下溢产生的中断请求。

0 = 禁止

1 = 使能

第 5 位 TCVIE：上溢中断使能位。使能或禁止由计数器上溢产生的中断请求。

0 = 禁止

1 = 使能

第 4 位 TPRIE：定时器周期寄存器中断使能位。使能或禁止由 TPR 寄存器比较匹配产生的中断请求。

0 = 禁止

1 = 使能

第 3 位 保留

第 2 位 TGCIE：定时器通用寄存器 C 中断使能位。使能或禁止由 TGRC 寄存器的输入捕获或比较匹配产生的中断请求。

0 = 禁止

1 = 使能

第 1 位 TGBIE：定时器通用寄存器 B 中断使能位。使能或禁止由 TGRB 寄存器的输入捕获或比较匹配产生的中断请求。

0 = 禁止

1 = 使能

第 0 位 TGAIE：定时器通用寄存器 A 中断使能位。使能或禁止由 TGRA 寄存器的输入捕获或比较匹配产生的中断请求。

0 = 禁止

1 = 使能

11.5.4 PDC 定时器的中断状态寄存器

中断状态寄存器指明了如下事件的发生：侦测位置改变，TCNT 下溢/上溢，周期寄存器比较匹配和 TGRA、TGRB、TGRC 的输入捕获/比较匹配等事件。这些标志指出了中断源。当 P_TMRx_INT (x = 0, 1)中相应中断使能位设置后，就可以产生中断。TCDF 位表示了在中心 PWM 模式或相位计数模式下计数器的计数方向。

P_TMR0_Status ($7425)：PDC 定时器 0 中断状态寄存器

P_TMR1_Status ($7426)：PDC 定时器 1 中断状态寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R/W
0	0	0	0	0	0	0	0

保留	PDCIF

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TCDF	TCUIF	TCVIF	TPRIF	保留	TGCIF	TGBIF	TGAIF

第 15:9 位　保留

第 8 位　PDCIF：侦测位置改变标志。该状态标志指明位置侦测寄存器内容是否有所变化。写入“1”可清除该标志。

0 ＝位置无变化

1 ＝位置有变化

第 7 位　TCDF：定时计数器方向标志，表示 TCNT 计数时的方向。

0 ＝递增计数

1 ＝递减计数

第 6 位　TCUIF：定时计数器下溢标志，表示在相位计数模式下 TCNT 是否发生数据下溢。写入“1”可清除该标志。

0 ＝未发生下溢

1 ＝发生下溢

第 5 位　TCVIF：定时计数器上溢标志，表示在相位计数模式下 TCNT 是否发生数据上溢。写入“1”可清除该标志。

0 ＝未发生上溢

1 ＝发生上溢

第 4 位　TPRIF：定时器周期寄存器比较匹配标志，表示 TPR 寄存器的比较匹配事件是否发生。写入“1”可清除该标志。

0 ＝未发生比较匹配事件

1 ＝发生比较匹配事件

第 3 位　保留

第 2 位　TGCIF：定时器通用寄存器 C 输入捕获/比较匹配标志，表示了 TGRC 寄存器输入捕获或者比较匹配事件是否发生。写入“1”可清除该标志。

0 ＝未发生输入捕获/比较匹配

1 ＝发生输入捕获/比较匹配

第 1 位　TGBIF：定时器通用寄存器 B 输入捕获/比较匹配标志，表示了 TGRB 寄存器输入捕获或者比较匹配事件是否发生。写入“1”可清除该标志。

0 ＝未发生输入捕获/比较匹配

1 ＝发生输入捕获/比较匹配

第 0 位　TGAIF：定时器通用寄存器 A 输入捕获/比较匹配标志，表示了 TGRA 寄存

器输入捕获或者比较匹配事件是否发生。写入“1”可清除该标志。

0 ＝未发生输入捕获/比较匹配

1 ＝发生输入捕获/比较匹配

11.5.5 定时器启动寄存器

P_TMR_Start 寄存器用来启动/停止 P_TMRx_TCNT (x = 0 - 4)的计数。一旦停止工作，P_TMRx_TCNT (x = 0 - 4)会自动清零。将 TMR0ST 或 TMR1ST 位写入 1 将会立即启动 P_TMR0_TCNT 或 P_TMR1_TCNT，反之写入 0 则立即停止 P_TMR0_TCNT 或 P_TMR1_TCNT。

P_TMR_Start ($7405)： 定时器计数启动寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
保留			TMR4ST	TMR3ST	TMR2ST	TMR1ST	TMR0ST

第 15:5 位 保留

第 4 位 **TMR4ST**：MCP 定时器 4 计数启动设置

0 ＝ 停止计数

1 ＝ 开始计数

第 3 位 **TMR3ST**：MCP 定时器 3 计数启动设置

0 ＝ 停止计数

1 ＝ 开始计数

第 2 位 **TMR2ST**：TPM 定时器 2 计数启动设置

0 ＝ 停止计数

1 ＝ 开始计数

第 1 位 **TMR1ST**：PDC 定时器 1 计数启动设置

0 ＝ 停止计数

1 ＝ 开始计数

第 0 位 **TMR0ST**：PDC 定时器 0 计数启动设置

0 ＝ 停止计数

1 ＝ 开始计数

11.5.6 PDC 定时器的侦测位置改变控制寄存器

在 SPMC75X 系列微控制器中有两个位置侦测控制寄存器：定时器 0 的 P_POS0_DectCtrl 和定时器 1 的 P_POS1_DectCtrl 。这两个寄存器控制了来自 TIOxA, TIOxB 和 TIOxC (x = 0, 1)引脚上的位置侦测信号采样的设定。采样参数包括采样时钟，有效采样计数选择和采样延时，这些都是可编程的。

SPLMOD 位决定着位置信号的采样条件，有以下三种模式可选择：PWM 有输出时采样、周期性采样、下相(UN, VN, WN)导通时采样。SPLCNT 位选择采样的延时，用于在采样完成但 PWM 仍有输出或下相(UN, VN, WN)导通的情况下，防止功率管导通的瞬态干扰造成的错误侦测。

P_POS0_DectCtrl ($7462)：PDC 定时器 0 位置侦测控制寄存器

P_POS1_DectCtrl ($7463)：PDC 定时器 1 位置侦测控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	RW	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SPLCK		SPLMOD		SPLCNT			

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
PDEN	SPDLY						

第 15:14 位　**SPLCK**：采样时钟选择位。

00 ＝ $F_{CK}/4$

01 ＝ $F_{CK}/8$

10 ＝ $F_{CK}/32$

11 ＝ $F_{CK}/128$

第 13:12 位　**SPLMOD**：采样模式选择位。

00 ＝ 在 Ux、Vx、Wx、UxN、VxN、WxN 有效并延迟 SPDLY 个计数周期后采样

01 ＝ 立即采样，不管采样延迟计数器 SPDLY 的设置

10 ＝ 下相(UxN, VxN, WxN)导通并延迟 SPDLY 个计数周期后采样

11 ＝ 保留

第 11:8 位　**SPLCNT**：采样次数设置。用来设置侦测位置信号时的采样次数。外部位置信号必须连续采样 SPLCNT 次才算作有效的位置信号，采样次数的有效值为从 1 到 15 次。注 意 0 和 1 都将被认为是 1 次。

第 7 位　**PDEN**：位置侦测使能位，用于使能/禁止来自 TIOA-C 管脚的位置输入信号的位置侦测功能。使能时，对这些引脚上的输入信号进行采样，将结果锁存入 POS_DectData 寄存器的 PDR [2:0]位中，一旦禁止该功能，PDR [2:0]中仍保持了先前的状态。

0 ＝ 禁止

1 ＝ 使能

第 6:0 位　**SPDLY**：采样延时计数设置。用于避免在 PWM 输出开启瞬间的噪声干扰导致的侦测错误。这个计数器在 Ux、Vx、Wx、UxN、VxN、WxN 有效后开始工作。

11.5.7 PDC 定时器的位置侦测数据寄存器

当前采集到的经过噪声滤波的位置侦测数据锁存在位置侦测数据寄存器里。可以在位置侦测控制寄存器 P_POSx_DectCtrl (x = 0, 1)里进行采样设定。

P_POS0_DectData ($7464)：PDC 定时器 0 位置侦测数据寄存器

P_POS1_DectData ($7465)：PDC 定时器 10 位置侦测数据寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留					PDR		

第 15:13 位　保留

第 3:0 位　PDR

PDR[2] ＝来自引脚 TIOOC 的经过噪声滤波的位置侦测数据输入。

PDR[1] ＝来自引脚 TIOOB 的经过噪声滤波的位置侦测数据输入。

PDR[0] ＝来自引脚 TIOOA 的经过噪声滤波的位置侦测数据输入。

11.5.8 PDC 定时器的计数寄存器

PDC 定时器有两个 16 位只读 TCNT 计数器(P_TMR0_TCNT 和 P_TMR1_TCNT)，可以根据输入时钟进行递增/递减计数，分别负责定时器 0 和定时器 1 的计数。

用相应的定时控制寄存器的 TMRPS 位用来选择输入时钟及分频，在中心 PWM 模式下 P_TMR0_TCNT 和 P_TMR1_TCNT 为递增/递减计数，在其他模式下，只能递增计数。TCNT 计数器重新清零继续递增计数，计数清除源包括 TGRA、TGRB、TGRC 和 P_TMR0_TPR 寄存器,或者侦测到在 P_POSx_DectData (x = 0 , 1)中有数据变化。当 TCNT 计数器发生上溢时，相应通道的中断状态寄存器中的 TCUIF 标志位将会置 1。

P_TMR0_TCNT ($7430)：PDC 定时器 0 计数寄存器

P_TMR1_TCNT ($7431)：PDC 定时器 1 计数寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0

TMRCNT

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
TMRCNT							

11.5.9 PDC 定时器的通用寄存器

TGRA、TGRB、TGRC 是 16 位寄存器。PDC 定时器有 6 个定时通用寄存器，定时器 0 和定时器 1 各占 3 个。TGR 寄存器为可读可写的 16 位双功能寄存器，既可以作为 PWM 输出也可以作为捕获输入。TGR 与 TCNT 的值不断的互相比较，当 TGR 寄存器用作 PWM 输出寄存器功能时 ，一旦两值相等，相应定时中断状态寄存器里的 TGAIF、TGBIF、TGCIF 位就会置 1。比较匹配的输出可以由 P_TMRx_IOCtrl （x= 0,1）中的 IOAMODE、IOBMODE 和 IOCMODE 来选择。当 TGR 寄存器用于输入捕获寄存器时，TCNT 的值会在外部信号侦测后存储下来，这时，相应定时中断状态寄存器里的 TGAIF、TGBIF、TGCIF 位就会置 1。捕获输入的侦测边沿可以由 TIOxA、TIOxB 和 TIOxC (x = 0, 1)进行选择并且可通过 P_TMRx_Ctrl (x = 0, 1)寄存器里的 CCLS 位进行编程。

在 PWM 模式下，无论是选择边沿 PWM 模式或中心 PWM 模式，TGR 寄存器都用作占空比值寄存器。复位时, TGR 寄存器初始值 0x0000。

当 CCLS 位设置成 100’b, 101’b, 110’b 时，PDC 计数器做位置捕获计数功能，主要用于驱动无刷直流电机的应用。霍尔位置信号连接到 TIOxA, TIOxB, TIOxC (x = 0, 1)管脚上。TCNT 寄存器的值根据 CCLS 的设置情况存入 TGRA 寄存器，并且 CLEGS 位应设置在双沿清除。当发生侦测位置改变事件时，TCNT 寄存器的值会被锁存入 TGRA，同时 TCNT 寄存器被复位为 0x0000。用户可以应用这个信息读出 TGRA 的正确值并计算出电机的速度。

P_TMR0_TGRA ($7440)：PDC 定時器 0 通用寄存器 A
P_TMR0_TGRB ($7441)：PDC 定時器 0 通用寄存器 B
P_TMR0_TGRC ($7442)：PDC 定時器 0 通用寄存器 C
P_TMR1_TGRA ($7443)：PDC 定時器 1 通用寄存器 A
P_TMR1_TGRB ($7444)：PDC 定時器 1 通用寄存器 B
P_TMR1_TGRC ($7445)：PDC 定時器 1 通用寄存器 C

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRGLR							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

0	0	0	0	0	0	0	0
TMRGLR							

11.5.10 PDC 定时器的缓冲寄存器

输入捕获寄存器由 TBRA、TBRB、TBRC 和 TGRA、TGRB、TGRC 构成双缓冲结构。PWM 比较匹配寄存器可被同步更新。

P_TMR0_TBRA ($7450)：PDC 定时器 0 缓冲寄存器 A

P_TMR0_TBRB ($7451)：PDC 定时器 0 缓冲寄存器 B

P_TMR0_TBRC ($7452)：PDC 定时器 0 缓冲寄存器 C

P_TMR1_TBRA ($7453)：PDC 定时器 1 缓冲寄存器 A

P_TMR1_TBRB ($7454)：PDC 定时器 1 缓冲寄存器 B

P_TMR1_TBRC ($7455)：PDC 定时器 1 缓冲寄存器 C

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
TMRBUF							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
TMRBUF							

11.5.11 PDC 定时器的周期寄存器

P_TMRx_TPR (x = 0, 1)是一个 16 位可读可写寄存器，用于设定定时器的计数周期，当 P_TMRx_TCNT (x = 0, 1)寄存器的计数达到 P_TMRx_TPR (x = 0, 1)寄存器的值时，P_TMRx_TCNT (x = 0, 1)寄存器会根据 P_TMRx_Ctrl (x = 0, 1)寄存器中的模式位清除计数器(递增计数模式)或者转为向下计数(连续递增/递减计数模式)。P_TMRx_TPR (x = 0, 1)的默认值是 0xFFFF，当它的值设定为 0x0000 时，P_TMRx_TCNT (x = 0, 1)计数器会停止计数并保持 0x0000 的值。

P_TMR0_TPR ($7435)：PDC 定时器 0 周期寄存器

P_TMR1_TPR ($7436)：PDC 定时器 1 周期寄存器

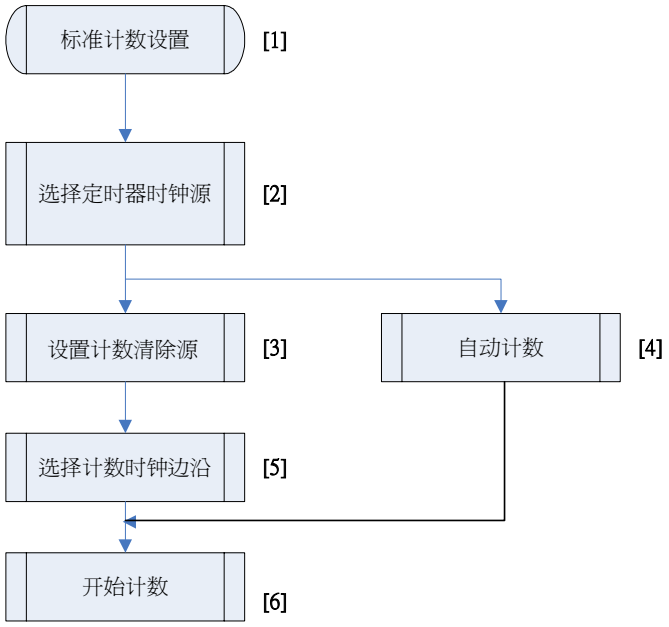
B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
TMRPRD							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
TMRPRD							

11.6 PDC 定时器的标准计数操作

P_TMRx_TCNT (x = 0, 1)的初始值是 0x0000，P_TMRx_TPR (x = 0, 1) 的初始值是 0xFFFF。当 TMR0ST 或 TMR1ST 位置 1，PDC 定时器相应的 P_TMRx_TCNT 寄存器就开始递增计数。计数器上溢后 P_TMRx_Status (x = 0, 1)中的 TCVIF 位会置 1，同时 PDC 定时器会产生一个中断请求。溢出后，P_TMRx_TCNT (x =0, 1)寄存器会重新从 0x0000 开始递增计数。

当 P_TMRx_TCNT (x =0, 1)寄存器选择一个周期匹配事件作为清除源的时候，计数器以周期性计数方式操作。通过将 CCLS 设定为 111'b，设置了 P_TMRx_TPR (x = 0, 1)寄存器的周期值和清除源。设置好后，当 TMR0ST 或 TMR1ST 位置 1 后，计数器开始做周期递增的操作，当计数值达到 P_TMRx_TPR (x = 0, 1)寄存器设置的值后，PDC 定时器中断状态寄存器的 TPRIF 位变为 1 且计数寄存器复位为 0x0000，如果 P_TMRx_INT (x = 0, 1)寄存器中的 TPRIE 位为使能，则发生 PDC 定时器中断请求。PDC 定时器标准计数操作的程序流程图如图 11-8 所示。



描述:

[1] 将控制寄存器中的 MODE 位设置为 0000'b。

[2] 通过 TMRPS 位选择定时器时钟源为外部时钟或内部时钟。

[3] 设置 P_TMRx_TPR (x = 0, 1) 并且 CCLS 位设置为 111'b。

[4] CCLS 采用默认设置 000'b，即禁止进行 TCNT 清除，计数器计数到 0xFFFF 溢出后重新从 0x0000 开始计数。

[5] 通过控制寄存器中的 CKEGS 位选择计数边沿。

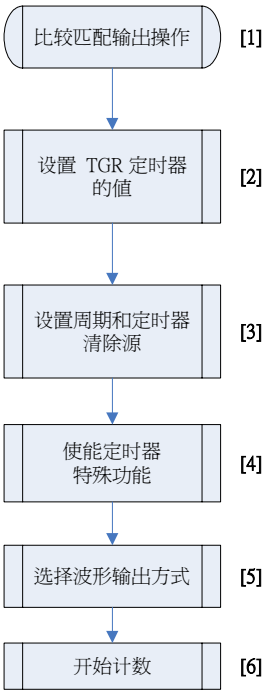
[6] 通过 P_TMR_Start 寄存器中的 TMR0ST 或 TMR1ST 的置位，启动计数器工作。

图 11-8 标准计数操作的程序流程图

11.7 PDC 定时器的 PWM 比较匹配输出操作

PDC 定时器有两个通道可完成总计 6 路 PWM 比较匹配输出功能。当(P_TMRx_TCNT (x =0, 1))与 P_TMRx_TGRA, P_TMRx_TGRB, P_TMRx_TGRC (x = 0, 1)寄存器比较匹配时，TIOxA, TIOxB, TIOxC (x = 0, 1) 引脚输出的 PWM 波形可以为高有效、低有效、输出保持。如图 11-9 所示为 PWM

比较匹配操作的编程流程图。图 11-10 描述了比较匹配输出和中断.



描述:

[1]设置 TGRA/TGRB/TGRC 的值，用于设定占空比。

[2] 将 CCLS 位设置成 111’b 确定计数周期为计数清除源 。

[3]将 P_IOB_SPE 寄存器中的 TIOOAEN, TIOOBEN, TIOOCEN 置 1，将相应的 IO 引脚配置成输出模式。

[4] 通过 P_TMRx_IOCtrl (x = 0, 1)寄存器选择比较匹配输出模式。

[5] P_TMR_Start 寄存器中的 TMR0ST 或 TMR1ST 置 1，计数器开始工作。

图 11-9 PWM 比较匹配输出操作的编程流程

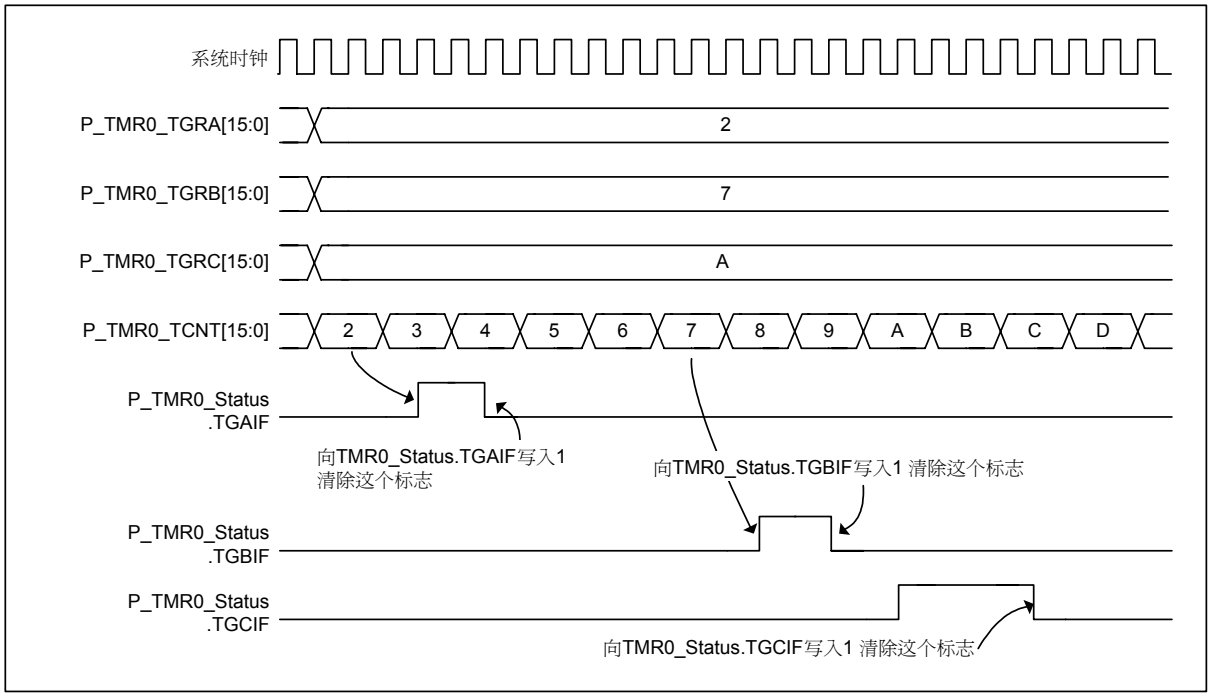


图 11-10 比较匹配输出和中断

11.8PDC 定时器的输入捕获操作

输入捕获操作可以对连接在 TIOxA、TIOxB、TIOxC（x=0、1）引脚上的信号进行测量，信号上升沿时计数寄存器 P_TMRx_TCNT 的值将被存入 P_TMRx_TGRA、P_TMRx_TGRB 或

P_TMRx_TGRC 中；下降沿时，计数寄存器 P_TMRx_TCNT 的值将被存入 P_TMRx_TBRA、P_TMRx_TBRB 或 P_TMRx_TBRC 中。

通过设置 P_TMRx_Ctrl (x = 0, 1)中的 CLEGS 位，来选择计数清除边沿，设置 CCLS 位来设置计数清除源，通过 P_TMRx_IOCtrl (x = 0, 1) 寄存器的 IOAMODE, IOBMODE 和 IOCMODE 位选择捕获边沿为上升沿、下降沿或者双沿。图 11-11 显示了捕获应用，信号连接 TIO0A。表 11-2 显示了输入捕获的设置和结果。

设置为输入捕获时，可以测量输入信号的脉宽和周期。如图 11-13 所示为输入捕获模式的编程流程图，图 11-12 所示为输入捕获信号脉宽和周期测量。

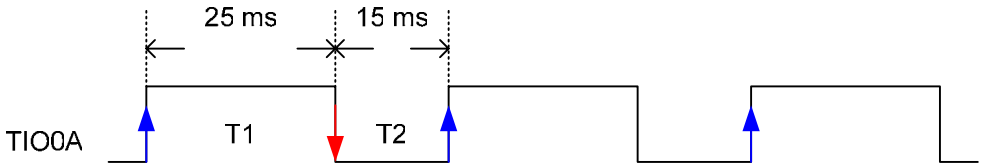


图 11-11 连接到 TIO0A 输入捕获信号

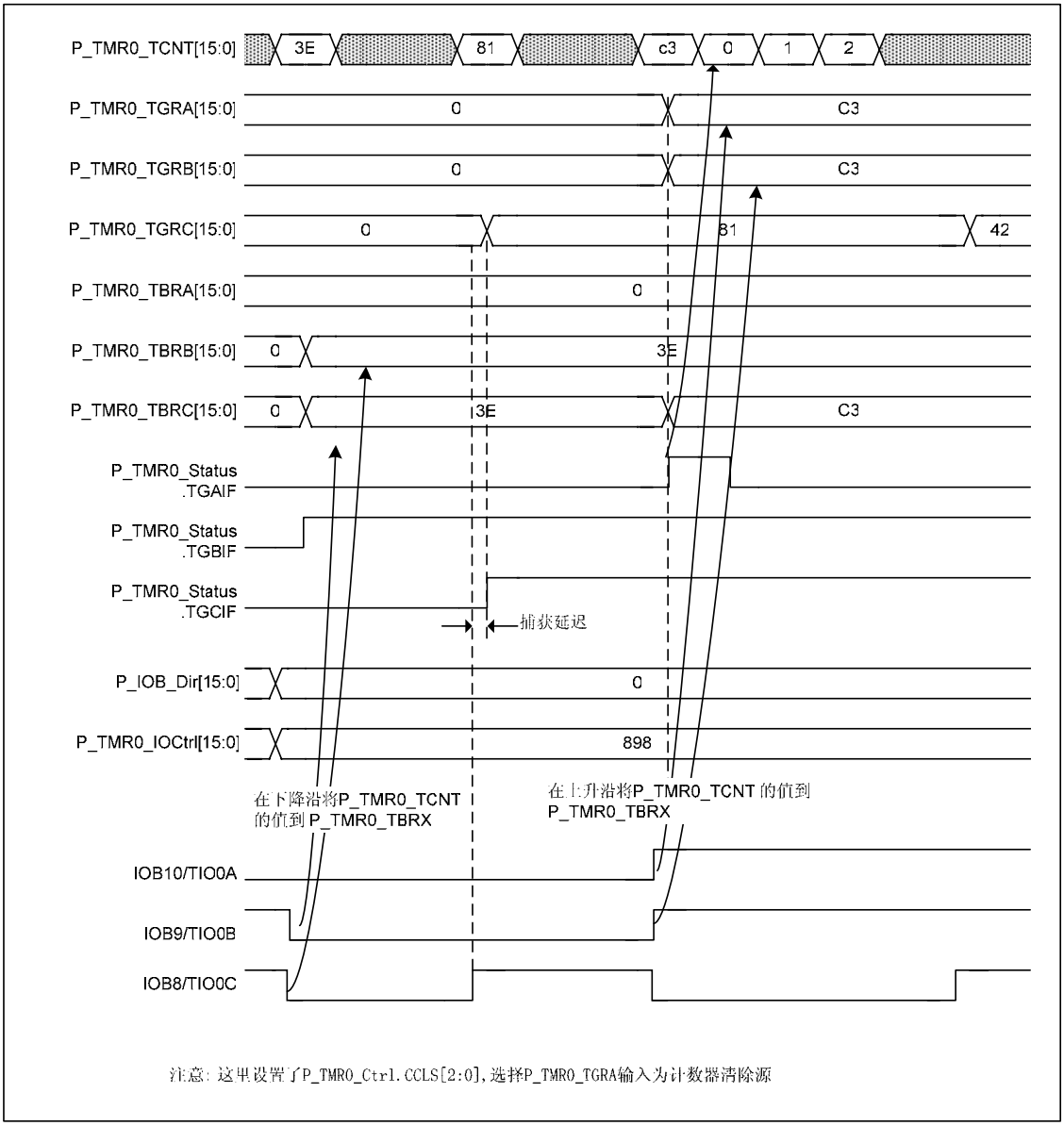


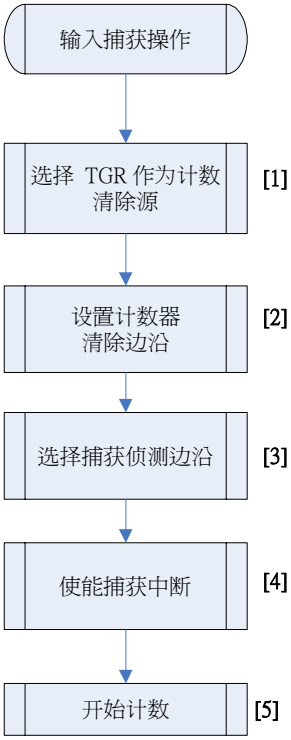
图 11-12　输入捕获信号脉宽和周期测量

表 11-2　捕获单路信号 的设置及结果

设置			结果	
清除沿 CLEGS	清除源 CCLS	捕获沿 IOAMODE	进中断沿	测量结果
上升沿	TGRA	上升沿	上升沿	P_TMR0_TGRA 周期 P_TMR0_TBRA 高电平宽度
上升沿	TGRA	下降沿	下降沿	P_TMR0_TGRA 周期 P_TMR0_TBRA 高电平宽度
上升沿	TGRA	双沿	双沿	P_TMR0_TGRA 周期；

设置			结果	
清除沿 CLEGS	清除源 CCLS	捕获沿 IOAMODE	进中断沿	测量结果
				P_TMR0_TBRA 高电平宽度
下降沿	TGRA	上升沿	上升沿	P_TMR0_TGRA 低电平宽度 P_TMR0_TBRA 周期
下降沿	TGRA	下降沿	下降沿	P_TMR0_TGRA 低电平宽度 P_TMR0_TBRA 周期
下降沿	TGRA	双沿	双沿	P_TMR0_TGRA 低电平宽度 P_TMR0_TBRA 周期
双沿	TGRA	上升沿	上升沿	P_TMR0_TGRA 低电平宽度 P_TMR0_TBRA 高电平宽度
双沿	TGRA	下降沿	下降沿	P_TMR0_TGRA 低电平宽度 P_TMR0_TBRA 高电平宽度
双沿	TGRA	双沿	双沿	P_TMR0_TGRA 低电平宽度 P_TMR0_TBRA 高电平宽度

如图 11-13 所示为输入捕获模式的编程流程图。



描述：

[1] 通过向 P_TMRx_Ctrl (x = 0, 1)寄存器的 CCLS 位写入设定的数值来选择 TGRA, TGRB, 或 TGRC 作为计数清除源。

[2] 通过向 P_TMRx_Ctrl (x = 0, 1)寄存器的 CLEGS 位写入设定的数值来设置计数清除边沿，来选择计数清除边沿为上升沿、下降沿或双沿。

[3] 通过设置 IOxMODE (x = A, B, C)位，来选择侦测捕获输入边沿为上升沿，下降沿或双沿。

[4]如有必要，在寄存器 P_TMRx_INT (x = 0, 1)中使能捕获中断。

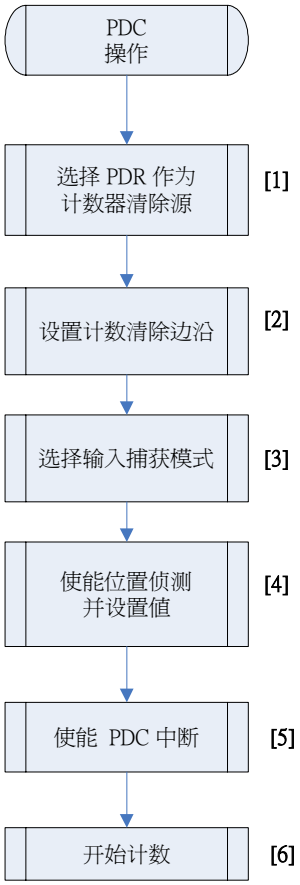
[5] P_TMR_Start 寄存器中的 TMR0ST 或 TMR1ST 置 1，计数器开始工作。

图 11-13 输入捕获操作的编程流程举例

11.9PDC 定时器的侦测位置改变(PDC)模式操作

PDC 定时器模块内部集成了 BLDC（直流无刷电机）驱动用的位置侦测传感器接口电路，为 BLDC 驱动提供了便利。当 P_TMRx_Ctrl (x = 0, 1)寄存器的 CCLS 位设置成 100'b, 101'b 或 110'b 时，在操作过程中 P_TMRx_TCNT (x = 0, 1)的值会自动传递给 TGRA ，这样寄存器 P_TMRx_TCNT (x = 0, 1)就可以将每隔 6 次、3 次或每次将位置数据都存入 TGRA。只要有侦测位置改变事件发生，寄存器 P_TMRx_TCNT (x = 0, 1)就会将数据传到 TGRA，将 PDCIF 中断标志位置 1，之后复位为 0x0000。如果在 P_TMRx_INT (x = 0, 1)中的位置侦测中断使能位 PDCIE 置 1，PDC 就会向 CPU 发出 PDC 中断请求。

通过对 SPLCNT, SPLCK 和 SPLMOD 位的值进行设置，可以避免在霍尔信号输入和位置侦测数据寄存器 P_POS0_DectData 中的噪声干扰，以使 P_POS0_DectData 能够锁存住正确的位置数据。PDC 模式的编程流程图如图 11-14 所示，噪声滤波位置侦测如图 11-15 所示。



描述：

[1] 通过将 P_TMRx_Ctrl (x = 0, 1)寄存器的 CCLS 位设置为 100’b, 101’b 或 110’b，.来选择位置数据改变作为计数清除源。

[2] 设置计数清除边沿为位置信号的双沿。将 P_TMRx_Ctrl (x = 0, 1)寄存器中的 CLEG 位设置为 11’b。

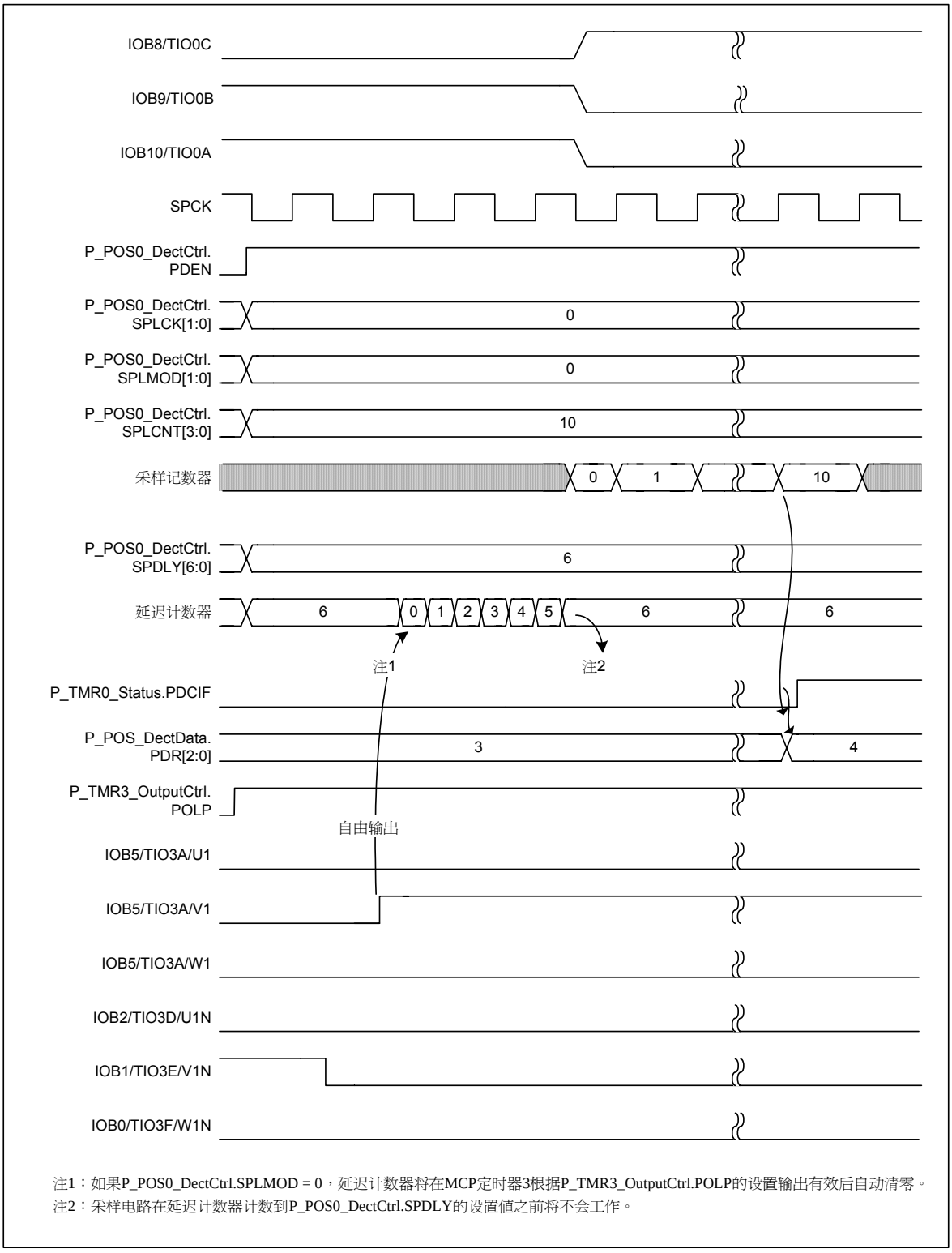
[3] 设置 P_TMRx_IOCtrl (x = 0, 1)寄存器，选择 PDR 模式，IOAMODE 应设置成 11xx’b.

[4] 通过设置 P_POSx_DectCtrl (x = 0, 1)寄存器中的 PDEN 位，使能位置侦测逻辑。同时，也要正确的设置 SPLMOD 和 SPLCNT 位的值

[5]] 如有必要，在寄存器 P_TMRx_INT (x = 0, 1)将 PDCIE 位置 1，使能 PDC 中断

[6] P_TMR_Start 寄存器中的 TMR0ST 或 TMR1ST 置 1，计数器开始工作。

图 11-14　PDC 定时器操作的编程流程举例



注1：如果P_POS0_DectCtrl.SPLMOD = 0，延迟计数器将在MCP定时器3根据P_TMR3_OutputCtrl.POLP的设置输出有效后自动清零。
注2：采样电路在延迟计数器计数到P_POS0_DectCtrl.SPDLY的设置值之前将不会工作。

图 11-15 位置侦测和噪声滤波时序

11.10 PDC 定时器的相位计数模式操作

SPMC75X 系列微控制器的 PDC 定时器支持四种相位计数模式。在相位计数模式中，如果两个外部时钟的输入不同将会被侦测出来，并且计数器会根据时钟相位关系递增或递减计数。该模式可以在 PDC 定时器 0 和 PDC 定时器 1 中设置，这种模式通常用于两相积分编码器脉冲输入。PDC 定时器 0 的时钟源是 TCLKA 和 TCLKB 引脚，PDC 定时器 1 的时钟源是 TCLKC 和 TCLKD 引脚。

图 11-16 到 11-19 所示为四种相位计数模式的操作时序。图 11-20 为相位计数模式的编程流程。

11.10.1 相位计数模式 1

在相位计数模式 1 下，当时钟源 TCLKB/TCLKD 超前 TCLKA/TCLKC90 度时，P_TMRx_TCNT (x = 0, 1)将是一直向上计数；当时钟源 TCLKB/TCLKD 滞后 TCLKA/TCLKC 90 度时，P_TMRx_TCNT (x = 0, 1)将是一直向下计数。这种模式适用于带编码器的电机的应用。表 11-3 列出了相位计数模式 1 的关系。相位的分辨率将解码器解析扩大 4 倍(脉冲/解析)。如图 11-16 所示为相位计数模式 1 的例子。

表 11-3 相位计数模式 1 的关系

TCLKA (PDC0) TCLKC (PDC1)	TCLKB (PDC0) TCLKD (PDC1)	计数操作
H	上升	递增
L	下降	
上升	L	
下降	H	
H	上升	递减
L	下降	
上升	H	
下降	L	

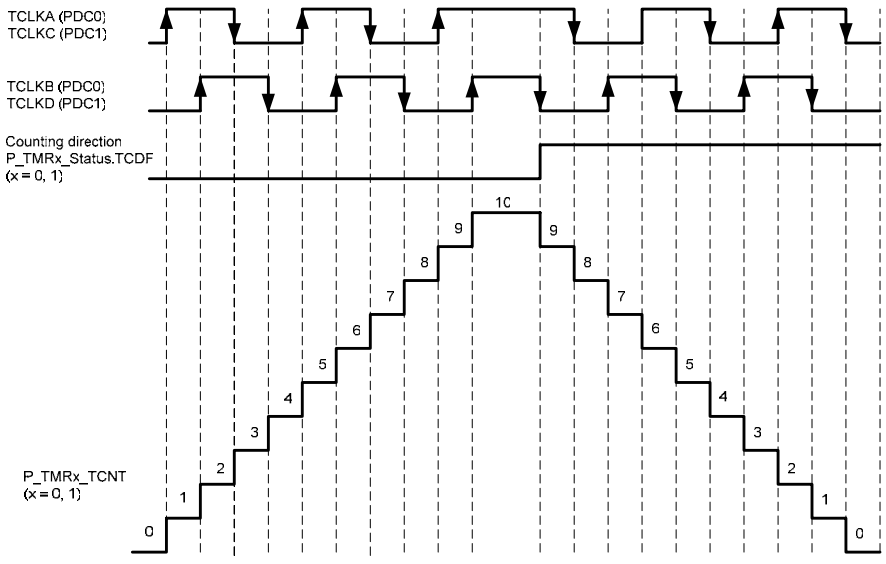


图 11-16 相位计数模式 1

11.10.2 相位计数模式 2

在相位计数模式 2 中，P_TMRx_TCNT (x = 0, 1)计数方向是由 TCLKB/TCLKD 的逻辑电平决定的。当 TCLKB/TCLKD 为逻辑高电平，计数器就向上计数；若 TCLKB/TCLKD 为逻辑低电平，计数器就向下计数。关系见表 11-4。计数器的动作与 TCLKA/TCLKC 的下降沿同步。图 11-17 为相位计数模式 2 的例子。

表 11-4 相位计数模式 2 关系

TCLKA (PDC0) TCLKC (PDC1)	TCLKB (PDC0) TCLKD (PDC1)	计数操作
H	上升	—
L	下降	—
上升	L	—
下降	H	递增
H	下降	—
L	上升	—
上升	H	—
下降	L	递减

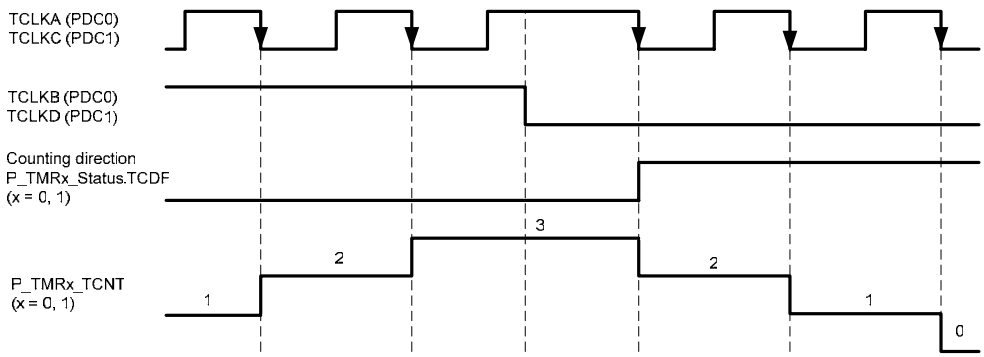


图 11-17 相位计数模式 2

11.10.3 相位计数模式 3

在相位计数模式 3 中，当 TCLKB/TCLKD 保持逻辑高电平时，在 TCLKA/TCLKC 的下降沿 P_TMRx_TCNT (x = 0, 1)递增计数；当 TCLKA/TCLKC 保持逻辑高电平状态，在 TCLKB/TCLKD 的下降沿 P_TMRx_TCNT (x = 0, 1)递减计数。关系见下表，图 11-18 为相位计数模式 3 的例子。

表 11-5 相位计数模式 3 关系

TCLKA (PDC0) TCLKC (PDC1)	TCLKB (PDC0) TCLKD (PDC1)	计数操作
H	上升	—
L	下降	—
上升	L	—

下降	H	递增
H	下降	递减
L	上升	—
上升	H	—
下降	L	—

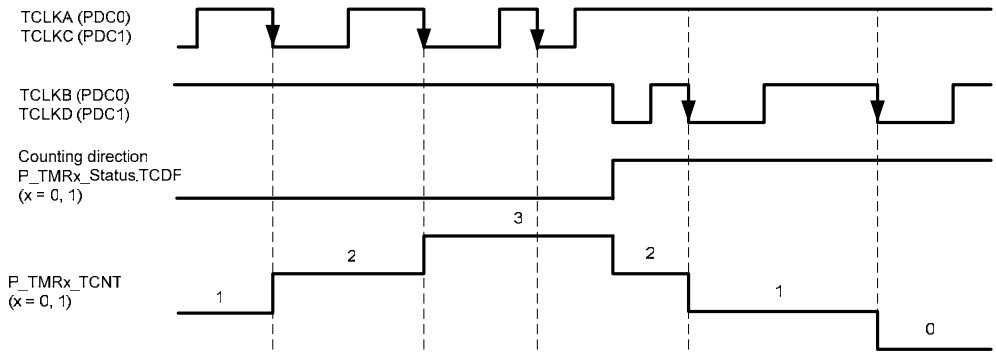


图 11-18 相位计数模式 3

11.10.4 相位计数模式 4

在相位计数模式 4 中，P_TMRx_TCNT (x = 0, 1)计数方向是由 TCLKx (x = A, B, C, D)的逻辑电平和边沿的选择联合决定的。当 TCLKx (x = A, C)为逻辑高/低电平，在 TCLKy (y = B, D)时钟上升/下降沿时触发计数器向上计数；当 TCLKx (x = A, C)为逻辑高/低电平，在 TCLKy (y = B, D)时钟下降/上升沿触发计数器向下计数。关系见下表 11-6。图 11-19 为相位计数模式 4 的例子。

表 11-6 相位计数模式 4

TCLKA (TPM 0) TCLKC (PDC1)	TCLKB (PDC0) TCLKD (PDC1)	计数操作
H	上升	递增
L	下降	递增
上升	L	—
下降	H	—
H	下降	递减
L	上升	递减
上升	H	—
下降	L	—

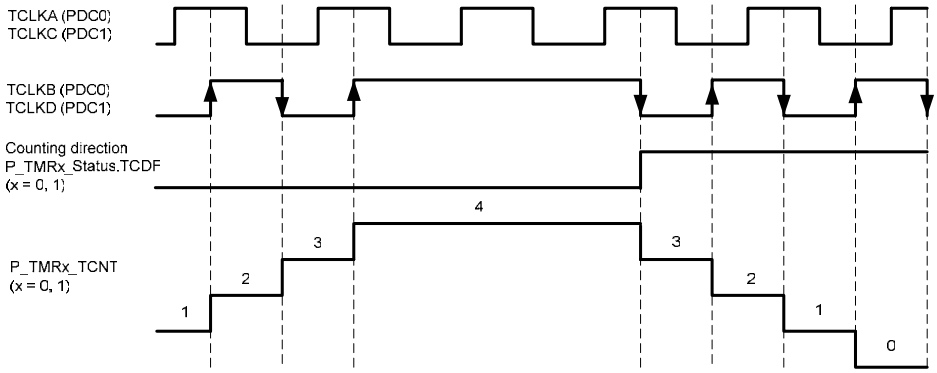
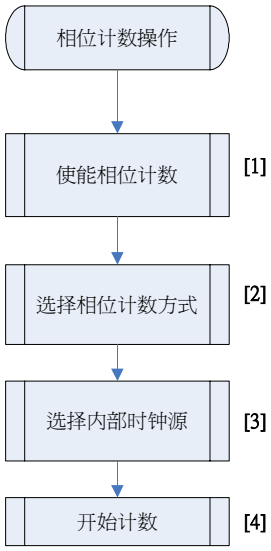


图 11-19 相位计数模式 4



描述：

[1] 通过设置 P_IOA_SPE 寄存器中 TCLKAEN/ TCLKBEN 或 TCLKCEN/ TCLKDEN 的值，使能相位计数模式。

[2] 通过设置 P_TMRx_Ctrl (x = 0, 1)寄存器中的 MODE 位，来选择相位计数模式 1-4。

[3] 通过设置 P_TMRx_Ctrl (x = 0, 1) 寄存器中的 TMRPS 正确的值，选择内部时钟源 FCK。

[4] P_TMR_Start 寄存器中的 TMR0ST 或 TMR1ST 置 1，计数器开始工作。

图 11-20 相位计数模式的操作流程编程举例

11.11 设计参考

【例 11-1】 下表演示了一个直流无刷电机驱动中 PDC 位置侦测和速度计算的编程过程。

```
/*****************************************************************************/
/*Init_PDC0():初始化 PDC0 模块并使能 PDC 中断*/
/* 侦测控制用来预防信号干扰*/
/*****************************************************************************/
void Init_PDC0(void)
{
 P_TMR0_Ctrl->W = CW_TMR0_TMRPS_FCKdiv64 | CW_TMR0_CCLS_PDR6 | \
                  CW_TMR0_CLEGS_Both | CW_TMR0_MODE_Normal;
                /* 上升沿计数, 每 6 次 PDR 变化清除一次 */
                                                /* 以 FCK/64 频率计数,上升沿 */
                                                /* 标准计数模式*/
P_TMR0_IOCtrl->W = CW_TMR0_IOAMODE_Capture_PDR;
P_TMR0_INT->W = CW_TMR0_PDCIE_ENABLE;          /* 使能 PDC0 中断 */
```

```
 /*====================================*/
 /* Position detection control 0 setup */
 /*====================================*/
P_POS0_DectCtrl->B.PDEN = 1;                          /*  使能侦测位置  */
P_POS0_DectCtrl->B.SPLMOD = 1;                        /*  周期采样  */
P_POS0_DectCtrl->B.SPLCNT = 15;
P_POS0_DectCtrl->W |= CW_TMR0_PDCR_SPLCK_FCKdiv128;   /* Init_PDC0() */
}

/*****************************************************************************/
/* IRQ1() : PDC 中断, 用于换流和速度计算*/
/*****************************************************************************/
void IRQ1(void) __attribute__ ((ISR));
void IRQ1(void)
{
 unsigned int n, h;

 if(P_TMR0_Status->B.PDCIF)
 {
        P_TMR0_Status->B.PDCIF = 1;
        n = P_TMR0_TGRA->W;

        if(n > P_TMR0_TCNT->W)
        {
                                                  /*连续 6 次 PDR 变化, 用于速度计算*/
                                                 /*用户可以应用  P_TMR0_TGRA 来完成*/
        }

h = P_POS0_DectData->W;                          /*  读当前霍尔信号  */
        ////////////////////////////////////
        // do BLDC motor phase commutation
        ////////////////////////////////////
 }
}                                                /* IRQ1() */
```

【例 11-2】 通过中断程序取得位置侦测数据并读取自霍尔传感信号输入的捕获计数器的值。位置侦测中断和捕获模块是两个不同且分别独立的外设功能。因此二者可以同时有效。捕获计数值只存储在 TGRA 中，同时向 CPU 发出中断请求。如下所示具体编码范例。

```
#include "Spmc75_regs.h"
#include "unspmacro.h"

int main(void)
{
```

```
disable_FIQ_IRQ();
Init_TMR3 ();                                   /* 初始化 MCP3 模块*/

P_IOB_SPE->W = 0x003F;                          /* 使能 UVW-1 输出特殊功能 */
P_TMR3_IOCtrl->W = 0x0111;                      /* TIO3A - TIO3F 输出使能 */
P_TMR3_OutputCtrl->B.DUTYMODE = 0;              /* 三相共用 TGRA 寄存器 */
P_TMR3_OutputCtrl->B.POLP = CB_TMR3_POLP_Active_High;          /* 高有效*/
P_POS0_DectCtrl->W = 0x0000;                    /* 采样时钟 = FCK/4 */
P_POS0_DectCtrl->B. SPLMOD = 1;                 /* 周期采样 */

/* TMR0 CCP special function Enable */
P_TMR0_INT->B.PDCIE = 1;                        /* 使能位置改变中断 */
P_POS0_DectCtrl->B.PDEN = 1;                    /* 使能位置侦测 */
P_POS0_DectCtrl->B.SPLMOD = 1;                  /* 周期采样*/

P_TMR0_IOCtrl->W = CW_TMR0_IOAMOD_Capture_PDR;
/*当 PDR 内容发生变化时进行输入捕获*/
P_TMR0_Ctrl->B.TMRPS = 1;                       /* 计数频率 FCK/4 */
P_TMR0_Ctrl->B.CCLS = 6;                        /* TCNT 由侦测位置改变清除 */
P_TMR0_Ctrl->B.CLEGS = 1;                       /* TCNT 在上升沿清除*/

P_TMR3_IOCtrl->W = 0x0111;
P_TMR_Start->W =    CW_TMR_TMR3ST_Start;        /* 启动 MCP3 */
P_TMR_Start->W |= CW_TMR_TMR0ST_Start;          /* 启动 PDC0*/
INT_FIQ_IRQ();                                  /* 使能 FIQ/IRQ 通道中断 */
}

/*****************************************************************************/
void IRQ1(void) __attribute__ ((ISR));
void IRQ1(void)
{
unsigned int cap_fifo;                          /* TGRA 寄存器的捕获值*/
unsigned int pdr_fido;                          /*位置侦测数据*/

if(P_INT_Status->B.PDC0IF)
{
if(P_TMR0_Status->B.PDCIF)
{
P_TMR0_Status->B.PDCIF = CB_TMR0_PDCIF_Enable;
pdr_fifo = P_POS0_DectData->W;
cap_fifo = P_TMR0_TGRA->W;
// 其他与 PDR 变化相关的任务
}
}
```

```
}
```

第12章 TPM 定时器 2 模块

12.1简介

SPMC75X 系列微控制器有一个 16 位通用定时器 TPM 定时器 2，支持捕获输入和 PWM 输出功能。定时器 2 为捕获输入和 PWM 输出操作提供两个输入/输出引脚。如图 12-1 所示为 TPM 定时器 2 的结构框图，详细规格见表 12-1。

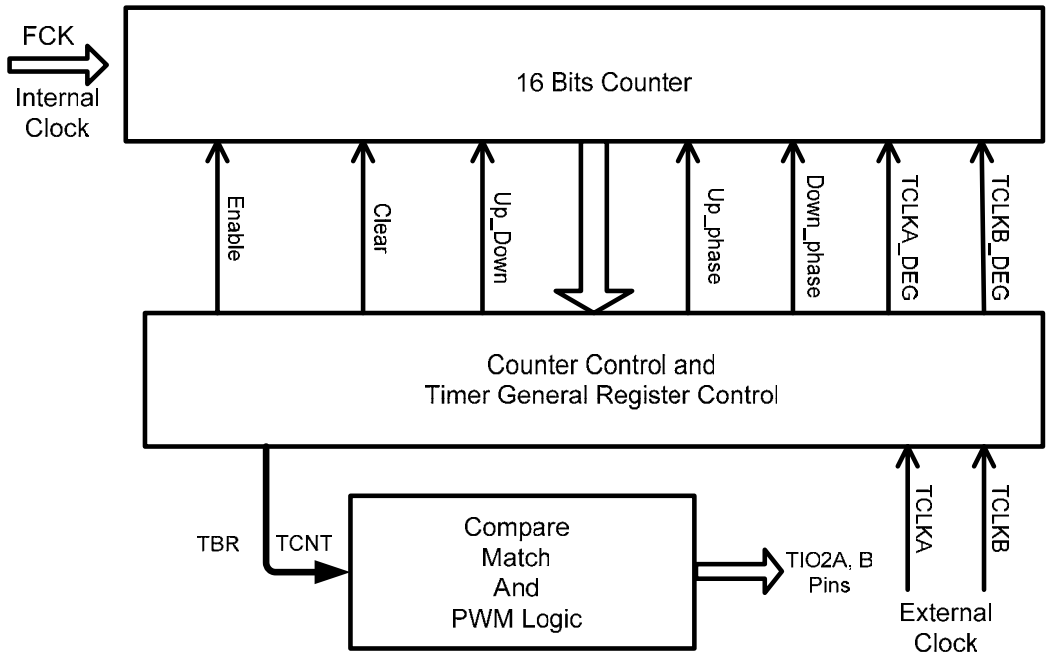


图 12-1　TPM 定时器 2 的功能示意框图

12.2定时器 2 的功能

- ❑ 能够处理两路捕获输入或PWM输出操作
- ❑ 有两个通用寄存器(P_TMR2_TGRA、P_TMR2_TGRB)：为PWM比较匹配输出或捕获输入之用
- ❑ 有两个定时缓冲寄存器(P_TMR2_TBRA、P_TMR2_TBRB)：用作PWM缓冲或捕获输入缓冲
- ❑ 8个可编程的时钟源：6个内部时钟(F_{CK}/1、F_{CK}/4、F_{CK}/16、F_{CK}/64、F_{CK}/256、F_{CK}/1024)，两个外部时钟(TCLKA和TCLKB)
- ❑ 可编程的定时器操作模式:

  1. 定时计数模式
  - 标准计数模式：连续递增计数
  - PWM比较匹配输出功能：在比较匹配和输出保持时既可以选择输出0也可以选择输出1
  - 输入捕获功能：选择由上升沿、下降沿、双沿(上升、下降)事件捕获和侦测位置改变触发

  2. PWM模式
  - 两个独立的PWM波形，以设定的占空比输出

3. 边沿PWM波形发生模式：
   – PWM以标准递增计数方式输出
4. 中心PWM波形发生模式：
   – PWM以递增/递减模式输出

- 共有3个中断源：
  – 一个定时器周期比较匹配中断源
  – 定时器通用寄存器P_TMR2_TGRA匹配中断源
  – 定时器通用寄存器P_TMR2_TGRB匹配中断源
- 定时器缓冲操作：
  – 输入捕获寄存器由TBRA、TBRB和TGRA、TGRB构成双缓冲结构。PWM 占空比寄存器(TGRA/B)会在周期中断时同步更新。
- 定时器的比较匹配和周期初始值可以任意设定
- 只读的16位递增/递减计数寄存器P_TMR2_TCNT。
- 16位可读写的定时周期寄存器：P_TMR2_TPR，提供系统时间基准。
- 两个可读写16位定时通用寄存器：P_TMR2_TGRA、P_TMR2_TGRB，用于输入捕获。
- 两个只读定时缓冲寄存器：P_TMR2_TBRA、P_TMR2_TBRB，是以上提到的定时通用寄存器的缓冲器。
- 16位可读写定时控制寄存器P_TMR2_Ctrl和输入输出控制寄存器P_TMR2_IOCtrl，后者是进行捕获输入或者PWM输出模式的选择。
- 16位可读写定时中断使能寄存器P_TMR2_INT，提供3个中断源，和一个16位只读中断状态寄存器P_TMR2_Status，它可记录中断的标志。

表 12-1 TPM 定时器 2 的规格

功能		TPM 定时器 2
时钟源		内部时钟： $F_{CK}/1$, $F_{CK}/4$, $F_{CK}/16$, $F_{CK}/64$、$F_{CK}/256$, $F_{CK}/1024$ 外部时钟： TCLKA, TCLKB
IO引脚		TIO2A, TIO2B
定时器通用寄存器		P_TMR2_TGRA, P_TMR2_TGRB
定时器缓冲寄存器		P_TMR2_TBRA, P_TMR2_TBRB
定时器周期与计数器寄存器		P_TMR2_TPR, P_TMR2_TCNT
捕获采样时钟		内部时钟：$F_{CK}/1$, $F_{CK}/2$, $F_{CK}/4$, $F_{CK}/8$
计数边沿		在上升沿、下降沿、双沿(上升、下降)计数
计数器清除源		在P_TMR2_TGRA, P_TMR2_TGRB捕获输入时清除 在P_TMR2_TPR比较匹配时清除
输入捕获功能		Yes
PWM比较匹配输出功能	输出1	Yes
	输出0	Yes
	输出保持	Yes
边沿PWM		Yes

功能	TPM 定时器 2
中心PWM	Yes
定时器缓冲操作	Yes
AD转换触发	P_TMR2_TGRA 比较匹配
中断源	定时器 2 TPR 中断、定时器 2 TGRA 中断、定时器 2 TGRB 中断

12.3TPM 定时器 2 的输入/输出/特殊功能引脚

通道	引脚名称	输入/输出	功能
外部时钟输入	TCLKA	输入	外部时钟 A 输入引脚
	TCLKB	输入	外部时钟 B 输入引脚
捕获输入 PWM 输出	TIO2A	输入/输出	P_TMR2_TGRA 输入捕获/PWM 比较匹配输出引脚
	TIO2B	输入/输出	P_TMR2_TGRB 输入捕获/PWM 比较匹配输出引脚

12.4TPM 定时器 2 的计数操作

TPM 定时器 2 是具有输入捕获和 PWM 比较匹配输出功能的通用定时器。TPM 定时器 2 有以下几种计数操作模式。

- ❑ 标准操作(标准递增计数)
- ❑ 依靠外部时钟输入 TCLKA 或 TCLKB 计数
- ❑ 边沿 PWM 模式(连续递增计数，PWM 输出模式)
- ❑ 中心 PWM 模式(递增/递减计数，PWM 输出模式)

P_TMR2_Ctrl 寄存器中的 MODE 位的值决定了 TPM 定时器 2 计数的模式。当定时器启动寄存器 P_TMR_Start 中的 TMR2ST 位置 1 时，定时器就从 0x0000 开始计数。并且 CCLS 位的值决定了计数器的清除源。

12.4.1 边沿 PWM 输出方式的连续递增计数模式

TPM 定时器 2 可由 P_TMR2_Ctrl 寄存器的 MODE 位设置为边沿 PWM 输出模式或没有任何 PWM 输出的标准操作模式。在此模式下，定时器从 0x0000 开始以递增方式计数，一直计数到与周期寄存器的设定值匹配为止。用户必须设定 P_TMR2_TPR 寄存器，并将计数清除源(CCLS)设置成由周期比较匹配来清除。

TPM 定时器进行连续递增计数, 其输入时钟源依据相应定时控制寄存器中 TMRPS 的值所决定。当定时器计数的值与定时器周期寄存器的值相等并且周期比较匹配中断标志 TPRIF 置 1 时，定时计数器清零。如果 P_TMR2_INT 中 PPRIE 使能，产生周期中断请求。当定时器计数器寄存器与 TGRA 或 TGRB 寄存器相匹配时，发生通用寄存器比较匹配事件。如果相应的定时器中断使能寄存器中的 TGAIE 或 TGBIE 使能，则产生通用寄存器比较匹配中断。

P_TMR2_TPR 的初始值可是从 0x0000 到 0xFFFF 之间的任何数值。定时器的时钟源既可以

选择外部时钟输入也可以选择内部时钟源 F_{CK}。标准连续递增计数模式非常适于数字电机控制系统中边沿触发或异步 PWM 和周期采样。如图 21-2 所示一个 TPM 定时器 2 的标准递增计数模式。

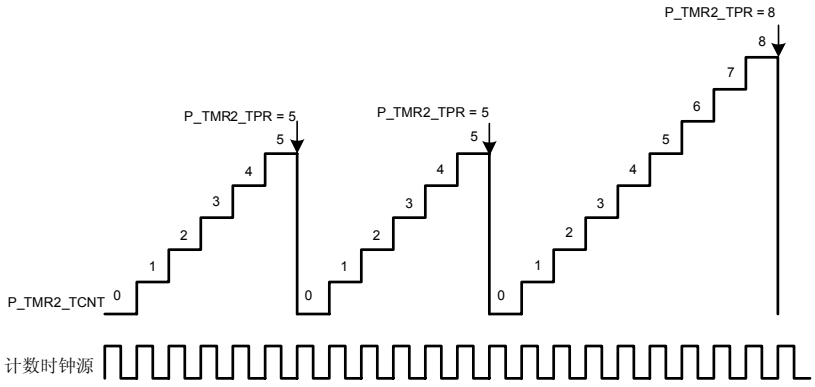


图 12-2 连续递增计数模式(CCLS = 111'b, CKEGS = 00'b, TMRPS = 000'b)

在边沿 PWM 模式下，用户需要设置 P_TMR2_TPR 周期寄存器和 P_TMR2_TGRx (x = A, B) 通用寄存器，然后将周期比较匹配事件设置成计数器清除源(CCLS)。比较匹配输出状态设置是在 P_TMR2_IOCtrl 寄存器中。定时器 2 的边沿 PWM 的标准连续递增计数模式如图 12-3 所示：如图 12-4 所示边沿 PWM 时序。

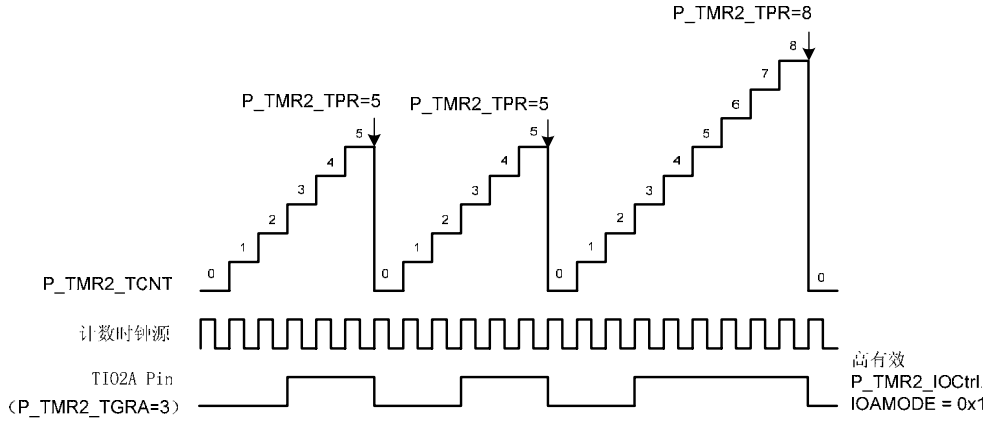


图 12-3 边沿 PWM 模式

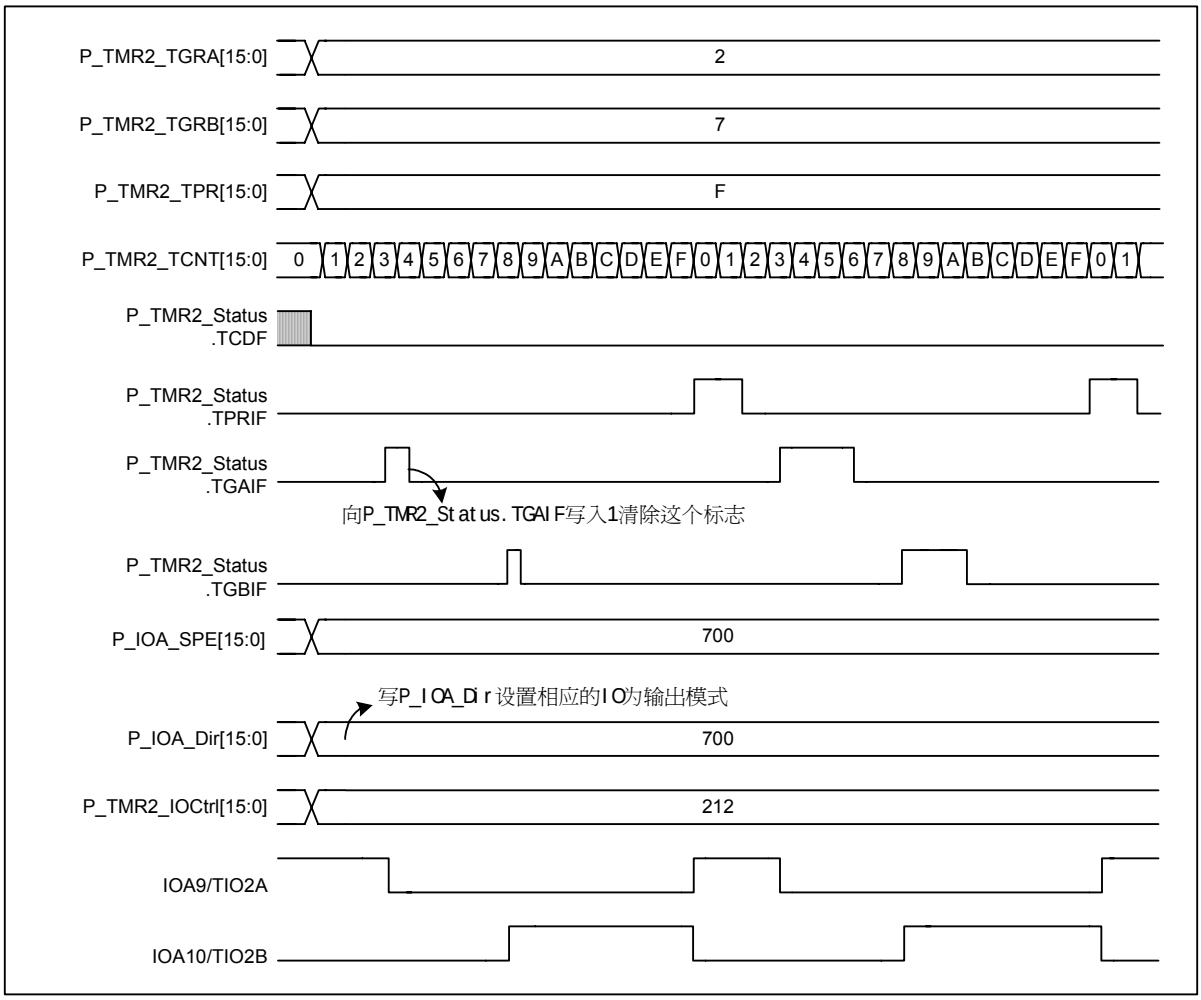


图 12-4 边沿 PWM 时序

12.4.2 中心 PWM 输出的连续递增/递减计数模式

连续递增/递减计数模式的操作与递增计数模式基本相同，唯一不同的是：定时器周期寄存器设定了整个计数过程的中间过渡点，计数过程中先递增，当达到定时周期寄存器设定值后开始递减。设置寄存器 P_TMR2_Ctrl 中 CKEGS 位，TPM2 定时器的周期是寄存器 P_TMR2_TPR 计算刻度的两倍。连续递增/递减计数模式的运行如图 12-5 所示。

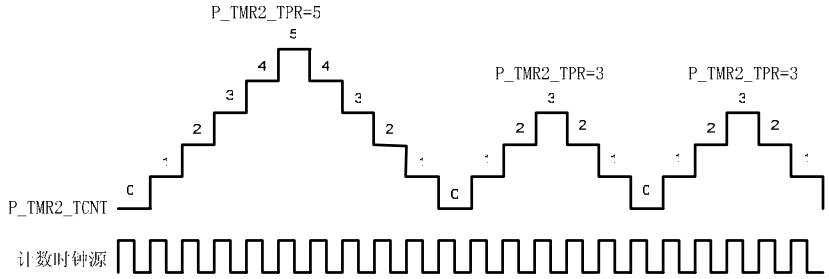


图 12-5 连续递增/递减计数模式(CCLS = 111'b, CKEGS = 00'b, TMRPS = 000'b)

TPM 定时器 2 周期寄存器的初始值可以是 0x0000 到 0xFFFF 中的任何数值，当定时计数寄存器的值与周期定时寄存器的值相等的时候，TPM 定时器 2 开始递减计数直到零。在这个过程中，周期中断的发生情况与递增计数模式的描述相同。计数的方向由 P_TMR2_Status 寄存器的 TCDF 位显示。定时器的时钟源既可以选择外部时钟输入也可以选择内部的 F_{CK} 时钟源。如图 12-6 所示为 TPM 定时器 2 在递增/递减计数模式下的中心 PWM 模式。图 12-7 所示中心 PWM 时序

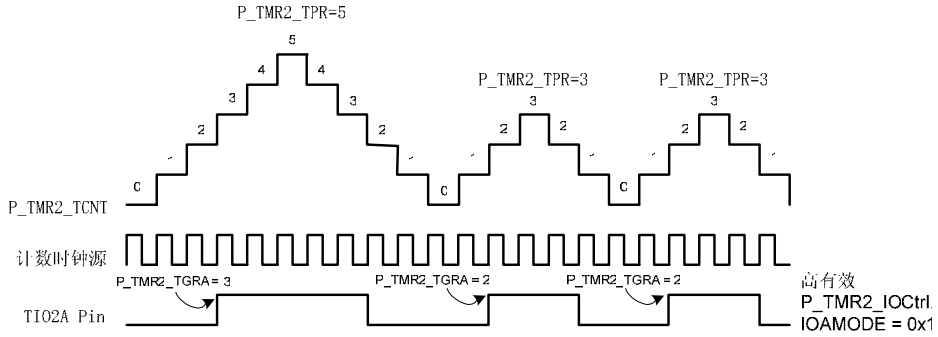


图 12-6 中心 PWM 模式

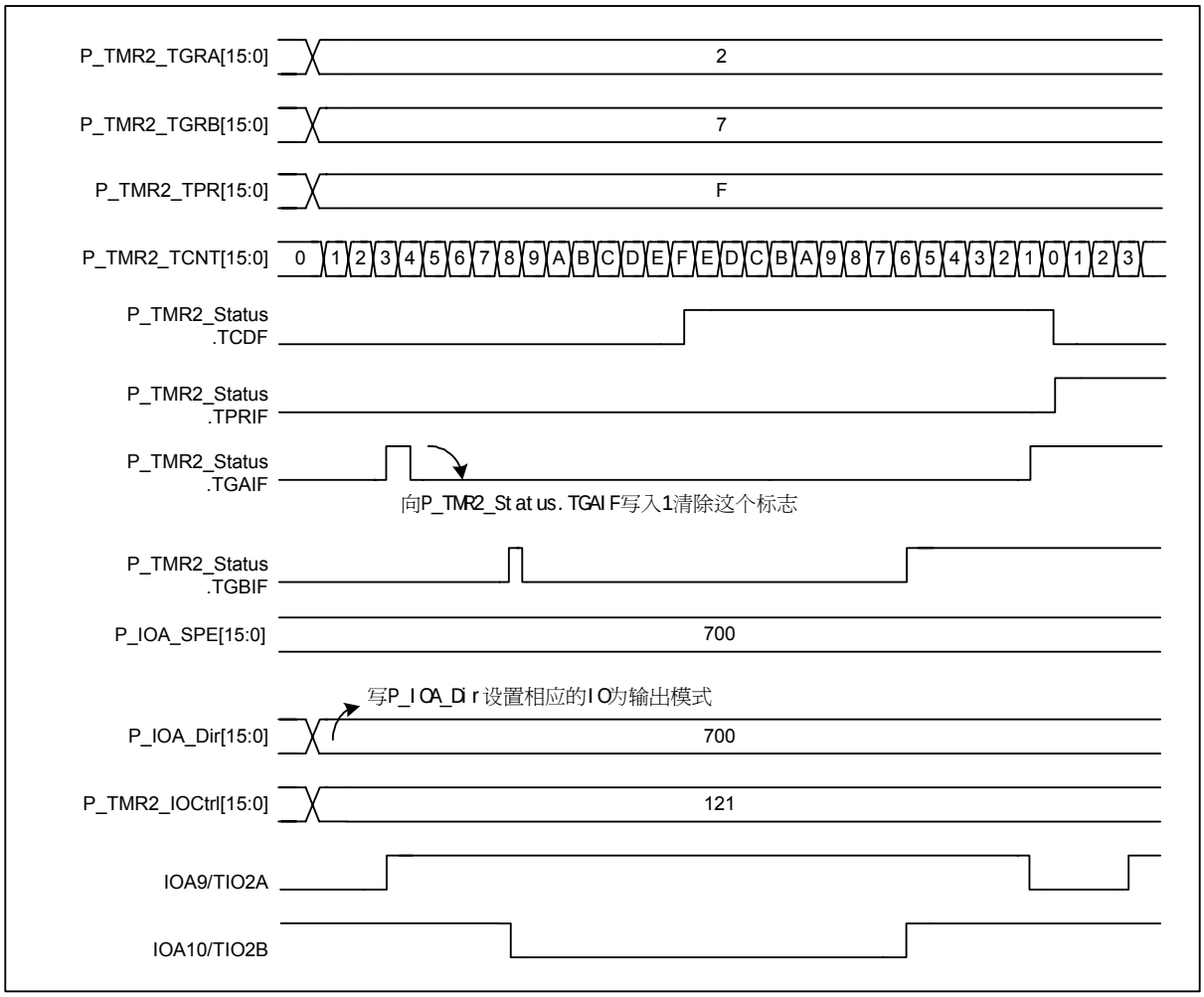


图 12-7 中心 PWM 时序

12.5定时器 2 控制寄存器

寄存器 P_TMR2_Ctrl 中配置了时钟源的选择、计数时钟边沿、计数器清除源、计数器清除边沿、输入捕获采样时钟和时钟操作模式等信息。TCLKA，TCLKB 时钟输入将由系统时钟 F_{CK} 采样。当设置为双沿触发计数时，输入时钟频率加倍。任何小于四倍采样时钟脉冲宽度的信号都会被忽略。

表 12-1 比较匹配定时器寄存器

地址	寄存器	名称
7402h	P_TMR2_Ctrl	定时器 2 控制寄存器
7412h	P_TMR2_IOCtrl	定时器 2 输入/输出控制寄存器
7422h	P_TMR2_INT	定时器 2 中断使能寄存器
7427h	P_TMR2_Status	定时器 2 中断状态寄存器
7432h	P_TMR2_TCNT	定时器 2 计数寄存器
7405h	P_TMR_Start	定时计数器启动寄存器
7446h	P_TMR2_TGRA	定时器 2 通用寄存器 A
7447h	P_TMR2_TGRB	定时器 2 通用寄存器 B
7456h	P_TMR2_TBRA	定时器 2 缓冲寄存器 A
7457h	P_TMR2_TBRB	定时器 2 缓冲寄存器 B
7437h	P_TMR2_TPR	定时器 2 周期寄存器

12.5.1 TPM 定时器 2 控制寄存器

P_TMR2_Ctrl ($7402)：TPM 定时器 2 控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SPCK		MODE				CLEGS	

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CCLS			CKEGS		TMRPS		

第 15:14 位 SPCK：捕获输入采样时钟选择位，用于选择捕获输入的采样时钟，小于采样时钟四倍频的频率将被滤掉。

00 = $F_{CK}/1$

01 = $F_{CK}/2$

10 = $F_{CK}/4$

11 = $F_{CK}/8$

第 13:10 位 MODE：模式选择位。用于选择定时器操作模式

0xxx = 标准模式(连续递增计数)

1x0x ＝ 边沿 PWM 模式(连续递增计数，PWM 输出)

1x1x ＝ 中心 PWM 模式(连续递增/递减计数，PWM 输出)

第 9:8 位　CLEGS：计数器清除边沿选择位，用于输入捕获模式下清除源的计数清除边沿触发方式。

00 ＝ 不清除

01 ＝ 上升沿

10 ＝ 下降沿

11 ＝ 上升/下降沿

第 7:5 位　CCLS：计数器清除源选择位，用于选择 TCNT 的计数清除源位置。

000 ＝ 禁止进行 TCNT 清除

001 ＝ 由 P_TMR2_TGRA 捕获输入清除 TCNT

010 ＝ 由 P_TMR2_TGRB 捕获输入清除 TCNT

011 ＝ 保留

100 ＝ 保留

101 ＝ 保留

110 ＝ 保留

111 ＝由 P_TMR2_TPR 比较匹配清除 TCNT

第 4:3 位　CKEGS：时钟边沿选择位，用于选择输入的时钟边沿触发方式。当输入时钟为双沿计数时，输入时钟周期会减半。若选择 $F_{CK}/1$ 为计数时钟，如果为双沿触发，那么计数器将以上升沿方式工作。

00 ＝ 上升沿计数

01 ＝ 下降沿计数

1X ＝ 双沿计数

第 2:0 位　TMRPS：定时器分频选择，用于选择 TCNT 计数时钟源。每个通道可分别的选择不同的时钟源。

000 ＝ $F_{CK}/1$ 下计数

001 ＝ $F_{CK}/4$ 下计数

010 ＝ $F_{CK}/16$ 下计数

011 ＝ $F_{CK}/64$ 下计数

100 ＝ $F_{CK}/256$ 下计数

101 ＝ $F_{CK}/1024$ 下计数

110 ＝ 以 TCLKA 引脚输入时钟计数

111 ＝ 以 TCLKB 引脚输入时钟计数

12.5.2　TPM 定时器 2 输入输出控制寄存器

P_TMR2_IOCtrl 寄存器控制 TIO2A 与 TIO2B 引脚上的 PWM 输出和捕获输入操作方式。通过设置位于 P_TMR2_Ctrl 寄存器里的 CCLS 位和 MODE 位可以确定定时器输入输出操作的模式。当

选择 PWM 比较匹配输出模式时，IOAMODE/IOBMODE 位决定着波形发生依赖于何种形式的时钟跳变沿。当选择捕获输入模式时，IOAMODE/IOBMODE 位对捕获事件进行了设置。

P_TMR2_IOCtrl ($7412) : TPM 定时器 2 输入输出控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
IOBMODE				IOAMODE			

第 15:8 位:　保留

第 7:4 位:　IOBMODE：　选择定时器 2 的 IOB 配置

第 3:0 位:　IOAMODE：　选择定时器 2 的 IOA 配置

PWM 比较匹配输出模式：

0000 ＝ 初始输出 0，当比较匹配时输出 0

0001 ＝ 初始输出 0，当比较匹配时输出 1

0010 ＝ 初始输出 1，当比较匹配时输出 0

0011 ＝ 初始输出 1，当比较匹配时输出 1

01xx ＝ 输出保持

输入捕获模式：

1000 ＝上升沿发生捕获事件中断

1001 ＝下降沿发生捕获事件中断

101x ＝ 双沿发生捕获事件中断

11xx ＝ 保留

12.5.3　TPM 定时器 2 的中断使能寄存器

P_TMR2_INT 寄存器用于使能或禁止以下中断请求：TGRA 寄存器发生比较匹配中断时是否进行 A/D 转换的中断请求、周期寄存器比较匹配的中断请求、TGRA 或 TGRB 的输入捕获/比较匹配的中断请求。

P_TMR2_INT ($7422)：TPM 定时器 2 的中断使能寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R	R	R/W	R	R	R/W	R/W
0	0	0	0	0	0	0	0
TADSE	保留		TPRIE	保留		TGBIE	TGAIE

第 15:8 位　保留

第 7 位　TADSE：A/D 转换启动请求使能位。使能或禁止当 TGRA 比较匹配完成后是否　启动 A/D 转换。

0 ＝ 禁止

1 ＝ 使能

第 6:5 位　保留

第 4 位　TPRIE：定时器周期寄存器中断使能位。使能或禁止 TPR 寄存器比较匹配需要的中断。

0 ＝ 禁止

1 ＝ 使能

第 3:2 位　保留

第 1 位　TGBIE：定时器通用寄存器 B 中断使能位。使能或禁止 TGRB 寄存器比较匹配　需要的中断。

0 ＝ 禁止

1 ＝ 使能

第 0 位　TGAIE：定时器通用寄存器 A 中断使能位。使能或禁止 TGRA 寄存器比较匹配　需要的中断。

0 ＝ 禁止

1 ＝ 使能

12.5.4　TPM 定时器 2 中断状态寄存器

中断状态寄存器表明了如下事件的发生：周期寄存器比较匹配和 TGRA 或 TGRB 的输入捕获/比较匹配输出等事件。这些标志表明了中断源。当 P_TMR2_INT 寄存器中相应中断使能位被置位后，就可以产生中断。TCDIF 位表明了在中心 PWM 模式下计数器的计数方向。

P_TMR2_Status ($7427)：TPM 定时器 2 中断状态寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R	R	R/W	R	R	R/W	R/W
0	0	0	0	0	0	0	0
TCDF	保留		TPRIF	保留		TGBIF	TGAIF

第 15:8 位　保留

第 7 位　TCDF：定时计数器方向标志，表示 TCNT 计数时的方向

0 ＝ 递增计数

1 ＝ 递减计数

第 6:5 位　保留

第 4 位　TPRIF：定时器周期寄存器比较匹配标志，表示 TPR 寄存器的比较匹配事件是否发生。写入“1”可清除该标志。

0 ＝ 未发生比较匹配事件

1 ＝ 发生比较匹配事件

第 3:2 位　保留

第 1 位　TGBIF：定时器通用寄存器 B 输入捕获/比较匹配标志，表示了 TGRB 寄存器输入捕获或者比较匹配事件是否发生。写入“1”可清除该标志。

0 ＝ 未发生输入捕获/比较匹配

1 ＝ 发生输入捕获/比较匹配

第 0 位　TGAIF：定时器通用寄存器 B 输入捕获/比较匹配标志，表示了 TGRA 寄存器输入捕获或者比较匹配事件是否发生。写入“1”可清除该标志。

0 ＝ 未发生输入捕获/比较匹配

1 ＝ 发生输入捕获/比较匹配

12.5.5　定时器计数启动寄存器

P_TMR_Start 寄存器用来启动/停止 P_TMR2_TCNT 的计数。一旦计数器停止工作，它的内容将会被清除。将 TMR2ST 置为 1 将会立即启动 P_TMR2_TCNT，反之写入 0 则立即停止 P_TMR2_TCNT。

P_TMR_Start ($7405)：定时器计数启动寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
保留			TMR4ST	TMR3ST	TMR2ST	TMR1ST	TMR0ST

第 15:5 位　保留

第 4 位　**TMR4ST**：MCP 定时器 4 计数启动设置

0 ＝ 停止计数

1 ＝ 开始计数

第 3 位　**TMR3ST**：MCP 定时器 3 计数启动设置

0 ＝ 停止计数

1 ＝ 开始计数

第 2 位　**TMR2ST**：TPM 定时器 2 计数启动设置

0 ＝ 停止计数

1 ＝ 开始计数

第 1 位　**TMR1ST**：PDC 定时器 1 计数启动设置

0 ＝ 停止计数

1 ＝ 开始计数

第 0 位　**TMR0ST**：PDC 定时器 0 计数启动设置

0 ＝ 停止计数

1 ＝ 开始计数

12.5.6　TPM 定时器 2 计数寄存器

TPM 定时器 2 有一个 16 位可读计数寄存器 P_TMR2_TCNT，可以根据输入时钟进行递增/递减计数。

相应定时器控制寄存器中 TMRPS 位可以选择输入时钟。在中心 PWM 模式下，P_TMR2_TCNT 可以递增/递减计数，在其他模式下，只能递增计数。当 P_TMR2_TCNT 寄存器与相应的 TGRA、TGRB 发生比较匹配，P_TMR2_TCNT 复位为 0x0000,或在输入捕获时，P_TMR2_TCNT 存入 TGRA、TGRB 。

P_TMR2_TCNT ($7432)：TPM 定时器 2 计数寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
TMRCNT							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
TMRCNT							

12.5.7　TPM 定时器 2 的通用寄存器

TGRA、TGRB 是 16 位寄存器。TPM 定时器 2 有 2 个定时器通用寄存器。TGR 寄存器为可读可写的 16 位双功能寄存器，既可以作为 PWM 输出也可以作为捕获输入。

当 TGR 寄存器用作 PWM 比较匹配输出寄存器时，TGR 与 TCNT 的值不断的互相比较，一旦两值相等，相应的定时器中断状态寄存器中的 TGAIF 或 TGBIF 位就被置 1。比较匹配的输出有 TIO2A 和 TIO2B。当 TGR 寄存器用于输入捕获寄存器时，在外部信号侦测后，TCNT 的值会存储下来，这时，相应的定时器中断状态寄存器中的 TGAIF 或 TGBIF 位就被置 1。捕获输入方式可以通过设置 TIO2A 或 TIO2B 的捕获边沿和通过设置 P_TMR2_Ctrl 寄存器里的 CCLS 位来进行。

在 PWM 模式下，无论是选择边沿 PWM 模式或中心 PWM 模式，TGR 寄存器都用作占空比值寄存器。通过复位，TGR 寄存器将初始化为 0x0000。

P_TMR2_TGRA ($7446)：TPM 定时器 2 通用寄存器 A

P_TMR2_TGRB ($7447)：TPM 定时器 2 通用寄存器 B

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRGLR							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRGLR							

12.5.8 TPM 定时器 2 的缓冲寄存器

输入捕获寄存器由 TBRA、TBRB 和 TGRA、TGRB 构成双缓冲结构。PWM 比较匹配寄存器可被自动更新。

P_TMR2_TBRA ($7456)：TPM 定时器 2 缓冲寄存器 A

P_TMR2_TBRB ($7457)：TPM 定时器 2 缓冲寄存器 B

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRBUF							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRBUF							

12.5.9 TPM 定时器 2 周期寄存器

P_TMR2_TPR 是一个 16 位可读可写寄存器，用于设定 PWM 波形的周期。当 P_TMR2_TCNT 寄存器的计数达到 P_TMR2_TPR 寄存器的值的时候，P_TMR2_TCNT 寄存器会根据 P_TMR2_Ctrl 寄存器中的模式位的编程清 0x0000(递增计数模式)或者转为向下计数(连续递增/递减计数模式)。P_TMR2_TPR 的默认值是 0xFFFF，当它的值设定为 0x0000 时，P_TMR2_TCNT 寄存器计数器会停止计数并保持 0x0000 的值。

P_TMR2_TPR ($7437)：TPM 定时器 2 周期寄存器

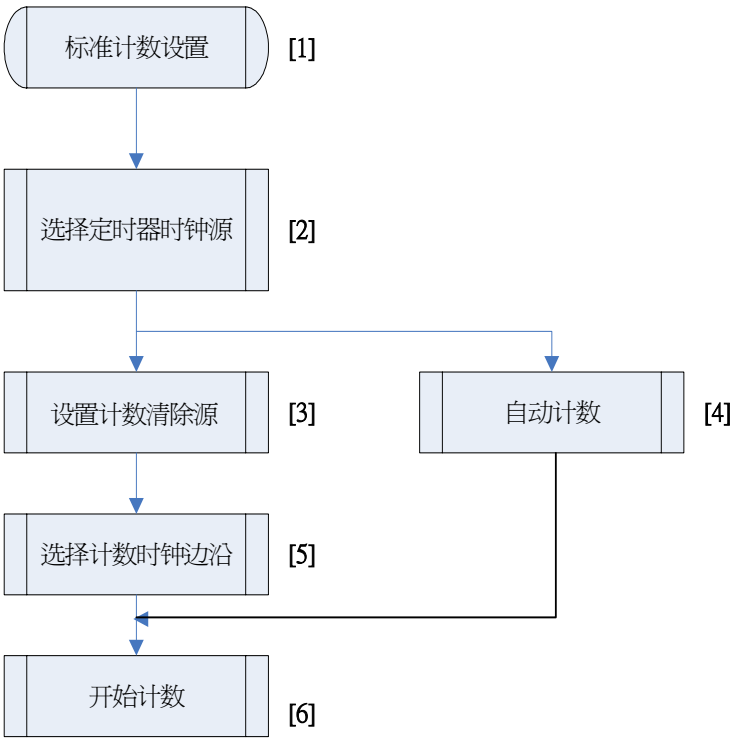
B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
TMRPRD							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
TMRPRD							

12.6 TPM 定时器 2 的操作

12.6.1 标准计数操作

当 P_TMR_Start 寄存器里的 TMR2ST 位置 1 后，定时器相应的 P_TMR2_TCNT 寄存器就开始递增计数。计数寄存器的工作方式如同自动计数的计数器，在计数递增到 0xFFFF 后复位为 0x0000。通过将 CCLS 位配置成 111’b，定时器被设置成周期性计数器。在定时寄存器达到 P_TMR2_TPR 寄存器中的数值时，将会复位为 0x0000。定时器 2 的标准计数操作的程序流程图如图 12-8 所示。



描述：

[1] 将定时器控制寄存器中的 MODE 位设为 0000’b

[2] 通过设定 TMRPS 位来选择定时器的内部或外部时钟源。

[3] 设定 P_TMR2_TPR 并将 CCLS 位设为 111’b。

[4] CCLS 采用默认设置 000’b，即禁止进行 TCNT 清除，计数器计数到 0xFFFF 溢出后重新从 0x0000 开始计数。

[5] 通过定时器控制寄存器中的 CKEGS 位来选择计数边沿。

[6] P_TMR_Start 寄存器中的 TMR2ST 位被设定后，开始计数。

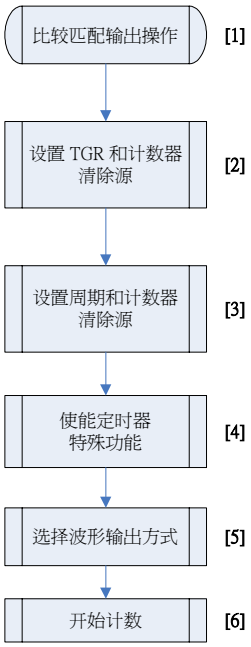
图 12-8 标准计数操作的程序流程图范例

P_TMR2_TCNT 的初始值是 0x0000，P_TMR2_TPR 的初始值是 0xFFFF 。当定时器开始位 TMR2ST 置 1 时，P_TMR2_TCNT 为自动计数自动计数。当计数器溢出时，P_TMR2_TCNT 寄存器从 0x0000 开始重新递增计数。

当选择了一个周期匹配事件作为 P_TMR2_TCNT 寄存器的清除源时，计数器为周期计数。用 P_TMR2_TPR 寄存器设置周期事件和通过设置 CCLS 位为 111'b 来设定计数清除源。这些设置好后，当定时器开始位 TMR2ST 置 1 后，计数器开始做周期递增的操作。当计数值达到 P_TMR2_TPR 寄存器设置的值后，定时器中断状态寄存器的 TPRIF 位被置 1，如果 P_TMR2_INT 寄存器 TPRIE 位被置 1 就发生一次定时器 2 的中断请求，计数器复位清零。

12.6.2 PWM 比较匹配输出操作

TPM 定时器 2 模块可完成共两路 PWM 比较匹配输出功能。当分别与 P_TMR2_TGRA 和 P_TMR2_TGRB 寄存器比较匹配时，相应的 TIO2A 和 TIO2B 输出引脚的输出的 PWM 波形有高有效、低有效、输出保持。图 12-9 为 PWM 比较匹配操作的编程流程图，图 12-10 为 PWM 比较匹配操作的中断时序。



描述：

[1] 设定 TGRA/TGRB 的值，用于产生设定的波形。

[2] 将 CCLS 位设为 111’b，从而周期寄存器确定周期和计数器清除源。

[3] 将 P_IOA_SPE 寄存器中的 TIO2AEN 和 TIO2BEN 位置 1，并将相应的 IO 引脚设为输出模式。

[4] 通过 P_TMR2_IOCtrl 寄存器来选择比较匹配输出模式。

[5] P_TMR_Start 寄存器中的 TMR2ST 被设置后，开始计数。

图 12-9 PWM 比较匹配输出操作的程序流程图范例

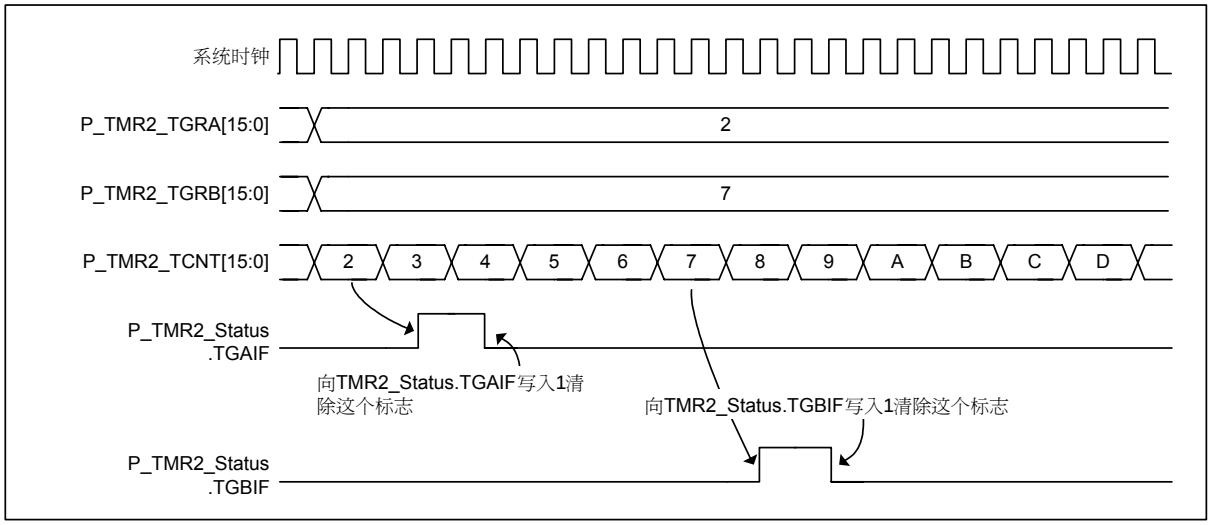


图 12-10　PWM 比较匹配输出中断时序

12.6.3　输入捕获操作

依据 P_TMR2_Ctrl 存器 CCLS 的设置，P_TMR2_TCNT 计数寄存器的值将被存入 TGRA、TGRB 中。通过 P_TMR2_IOCtrl 寄存器的 IOAMODE, IOBMODE 位选择捕获边沿为上升沿、下降沿或者双沿。

图 12-11 所示为 TIO2A 输入捕获信号。表 12-2 描述了输入捕获的设置及结果。如图 12-13 所示为输捕获模式的编程流程图，图 12-12 所示为输入捕获信号脉宽和周期测量时序。

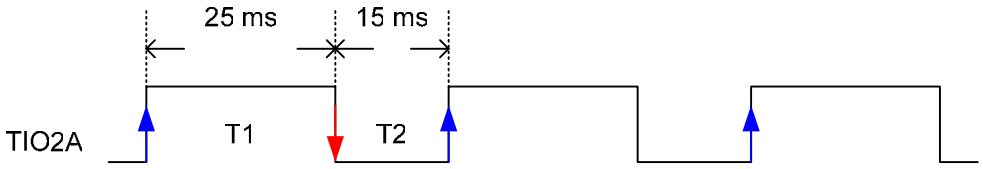


图 2-11　TIO2A 输入捕获信号

表 12-2　捕获单路信号 的设置及结果

设置			结果	
清除沿 CLEGS	清除源 CCLS	捕获沿 IOAMODE	进中断沿	测量结果
上升沿	TGRA	上升沿	上升沿	P_TMR2_TGRA 周期; P_TMR2_TBRA 高电平宽度
上升沿	TGRA	下降沿	下降沿	P_TMR2_TGRA 周期; P_TMR2_TBRA 高电平宽度
上升沿	TGRA	双沿	双沿	P_TMR2_TGRA 周期; P_TMR2_TBRA 高电平宽度

下降沿	TGRA	上升沿	上升沿	P_TMR2_TGRA 低电平宽度； P_TMR2_TBRA 周期
下降沿	TGRA	下降沿	下降沿	P_TMR2_TGRA 低电平宽度； P_TMR2_TBRA 周期
下降沿	TGRA	双沿	双沿	P_TMR2_TGRA 低电平宽度； P_TMR2_TBRA 周期
双沿	TGRA	上升沿	上升沿	P_TMR2_TGRA 低电平宽度； P_TMR2_TBRA 高电平宽度
双沿	TGRA	下降沿	下降沿	P_TMR2_TGRA 低电平宽度； P_TMR2_TBRA 高电平宽度
双沿	TGRA	双沿	双沿	P_TMR2_TGRA 低电平宽度； P_TMR2_TBRA 高电平宽度

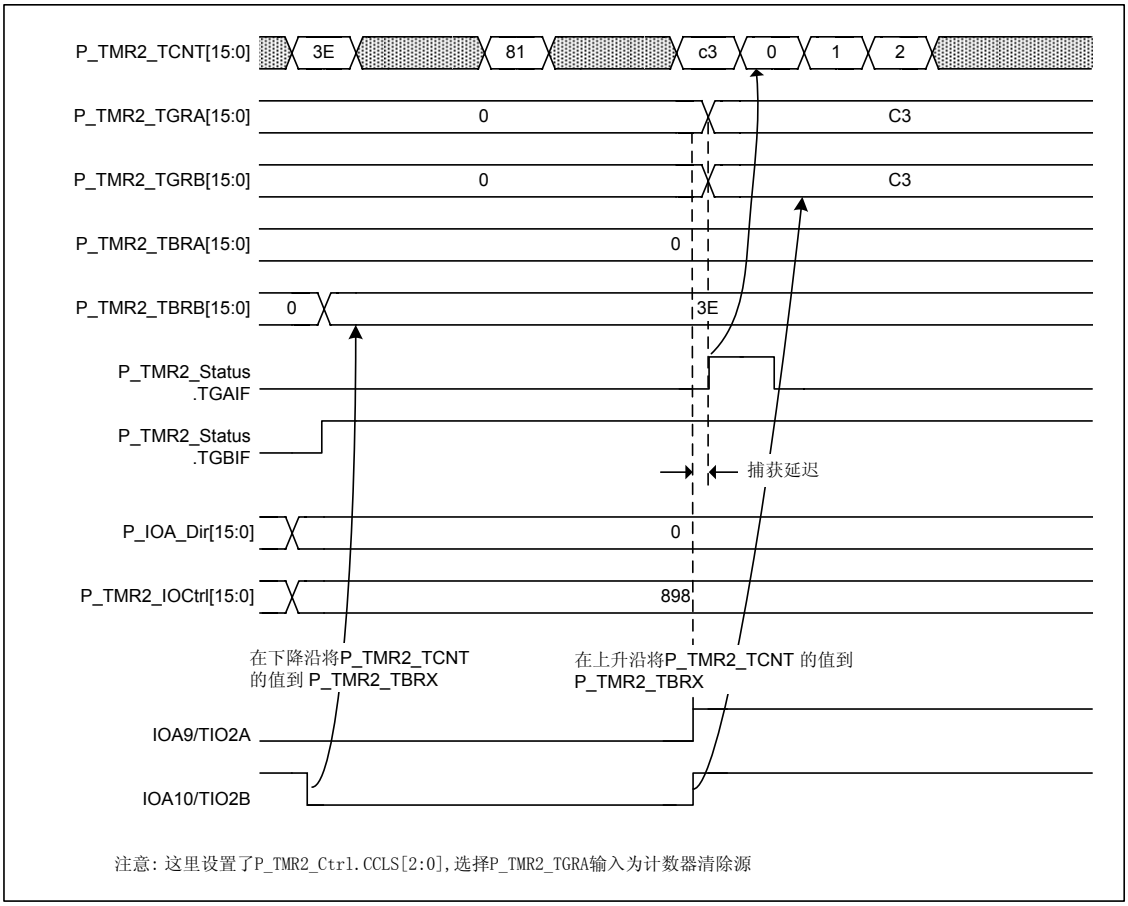
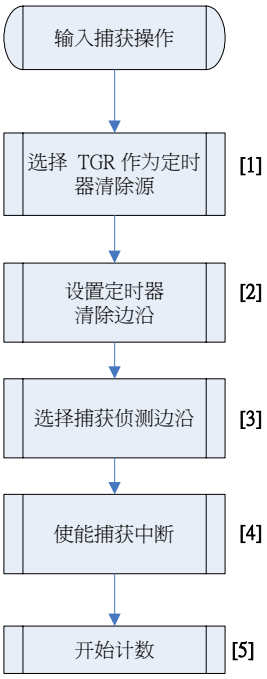


图 12-12 输入捕获信号脉宽和周期测量时序

输入捕获模式的编程流程图如图 12-13 所示。



描述：

[1] 通过向 P_TMR2_Ctrl 寄存器的 CCLS 位写入设定的数值来选择 TGRA, 或 TGRB 作为计数清除源

[2] 通过向 P_TMR2_Ctrl 寄存器的 CLEGS 位写入设定的数值来设置计数清除边沿，来选择计数清除边沿为上升沿、下降沿或双沿。

[3] 通过设定 IOxMODE (x = A, B)位来选择捕获输入侦测边沿为上升沿、下降沿或双沿。

[4] 如必要，将 P_TMR2_INT 寄存器中的捕获中断使能。

[5] P_TMR_Start 寄存器中的 TMR2ST 被置位后，开始计数。

图 12-13 输入捕获操作的程序流程图范例

12.7 设计参考

【例 12-1】下面给出了如何产生定时器 2 的周期中断和 P_TMR2_TGRA、 P_TMR2_TGRB 比较匹配中断的例子。

```
#include "Spmc75_regs.h"
#include "Spmc_typedef.h"

void Init_TMR2(void)
{
P_TMR2_Ctrl->W = CW_TMR2_CKEGS_Rising | CW_TMR2_CCLS_TPR;
P_TMR2_Ctrl->W |= CW_TMR2_MODE_PWM_Edge;

P_TMR2_TPR->W = 24000000 / 4000;                /* PWM 载波频率. = 4000 Hz */
P_TMR2_Ctrl->W |= CW_TMR2_TMRPS_FCKdiv1;
P_IOA_SPE->B.TIO2AEN = 1;
P_IOA_SPE->B.TIO2BEN = 1;

P_IOA_Dir->B.bit9  = 1;
P_IOA_Dir->B.bit10 = 1;

P_TMR2_IOCtrl->W = CW_TMR2_IOAMOD_Output_01 | CW_TMR2_IOBMOD_Output_01;
P_TMR2_INT->W = CW_TMR2_TPRIE_Enable;
P_TMR2_INT->W |= (CW_TMR2_TGAIE_Enable | CW_TMR2_TGBIE_Enable);
```

```
P_TMR_Start->B.TMR2ST = 1;
}                                                    /* Init_TMR2() */

/*****************************************************************************/
/* IRQ4() :中断源为 TPM2IF 的 IRQ4 中断服务子程序*/
/*****************************************************************************/
void IRQ4(void) __attribute__ ((ISR));
void IRQ4(void)
{
if(P_TMR2_Status->B.TPRIF)
{
P_TMR2_Status->B.TPRIF = 1;
TPM2_TPRINT_ISR();
        }

        if(P_TMR2_Status->B.TGAIF)
        {
                P_TMR2_Status->B.TGAIF = 1;
                TPM2_TGRAINT_ISR();
        }

        if(P_TMR2_Status->B.TGBIF)
        {
                P_TMR2_Status->B.TGBIF = 1;
                TPM2_TGRBINT_ISR();
        }
}                                                    /* IRQ4() */

/*****************************************************************************/
/* TPM2_TPRINT_ISR() :  定时器 2 周期中断*/
/*****************************************************************************/
void TPM2_TPRINT_ISR(void)
{
P_TMR2_TGRA->W = (5    * P_TMR2_TPR->W) / 100;        /* 5 %  占空比  */
P_TMR2_TGRB->W = (95L * P_TMR2_TPR->W) / 100;        /* 95 %占空比  */
} /* TPM2_TPRINT_ISR() */

/*****************************************************************************/
/* TPM2_TGRAINT_ISR() : TPM2 general register A compare match interrupt */
/*****************************************************************************/
void TPM2_TGRAINT_ISR(void)
{
 P_TMR2_TGRA->W = (5    * P_TMR2_TPR->W) / 100;       /* 5 %占空比*/
} /* TPM2_TGRAINT_ISR */
```

```
/*****************************************************************************/
/* TPM2_TGRBINT_ISR() : TPM2 general register B compare match interrupt */
/*****************************************************************************/
void TPM2_TGRBINT_ISR(void)
{
P_TMR2_TGRB->W = (95L * P_TMR2_TPR->W) / 100;                    /* 95 %占空比*/
}                                                                /* TPM2_TGRBINT_ISR */
```

第13章 MCP 定时器模块

13.1简介

SPMC75X 系列微控制器提供了两个 MCP(Motor Control PWM)定时器，定时器 3 和定时器 4。MCP 定时器有两套独立的三相六路 PWM 波形输出。MCP 定时器 3 与 PDC 定时器 0 联合，MCP 定时器 4 与 PDC 定时器 1 联合能完成无刷直流电机和交流感应电机应用中的速度反馈环控制。MCP 模块有总计 12 路定时器输出用作电机控制操作。如图 13-1 所示为 MCP 定时器 3 和定时器 4 的整体框图。详细规格说明见表 13-1。

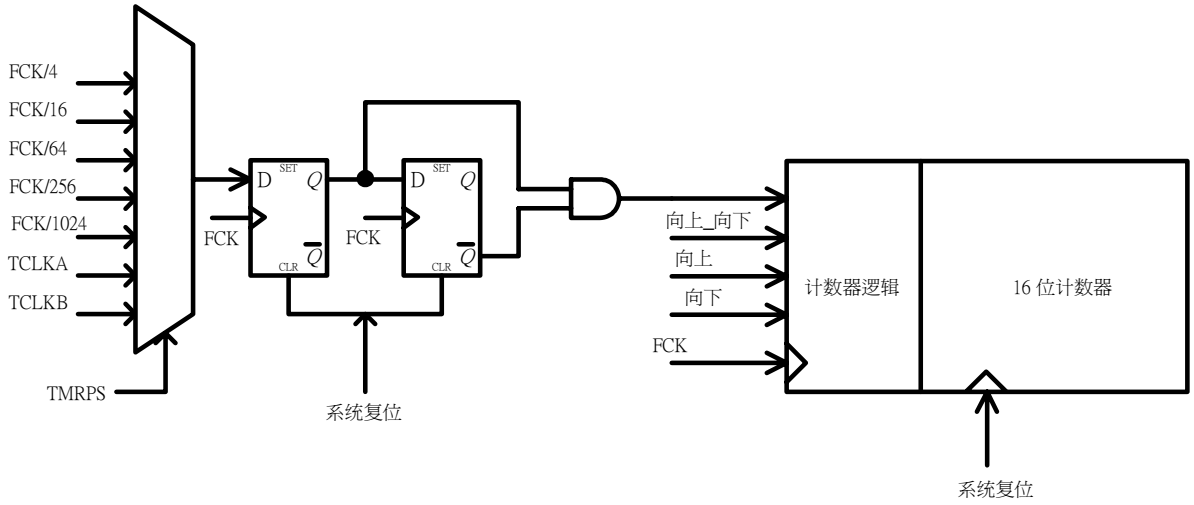


图 13-1　MCP 定时器 3 和定时器 4 结构框图

13.2MCP 定时器 3 和 MCP 定时器 4 的特性

- 能够产生12路可编程的PWM波形
- 6个通用定时寄存器(TGRAx/TGRBx/TGRCx, x = 3, 4)：每个定时器有3个独立的寄存器，用于 PWM比较匹配输出功能
- 6个定时缓冲寄存器(TBRAx/TBRBx/TBRCx, x = 3, 4)：每个定时器有3个独立的缓冲寄存器，负责PWM输出缓冲操作
- 为了防止PWM占空比载入不完整产生的问题：提供一个载入控制寄存器保证旋波PWM占空比值可以同时载入
- 可选择8个可编程的时钟源：6个内部时钟(F_{CK}/1, F_{CK}/4, F_{CK}/16, F_{CK}/64, F_{CK}/256, F_{CK}/1024)，两个外部时钟(TCLKA和TCLKB)
- 可编程的定时器操作模式：
  1. 定时器计数模式
     - 标准计数模式：递增计数
     - PWM比较匹配功能：在比较匹配时和输出保持时，可以选择输出0或1

2. PWM模式
   - 两套独立可设置占空比的6路PWM输出
3. 边沿PWM发生模式
   - PWM输出，以标准和递增计数方式操作
4. 中心PWM发生模式
   - PWM以递增/递减模式输出

- 共10个中断源，每个定时器5个
  - 定时周期比较匹配中断
  - TGRD寄存器比较匹配中断源
  - 外部错误输入中断
  - 外部过载输入中断
  - PWM输出短路保护中断
- UVW相位输出同步源选择：与位置数据寄存器P_POSx_DectData (x = 0, 1)变化同步，与TGRB或 TGRC寄存器的比较匹配同步
- PWM 占空比选择：共同利用TGRA或者三个相位各自独立选择
- 定时器缓冲操作：
  - PWM比较匹配寄存器可自动更新
- 比较匹配定时器周期初始值可任意设定
- 只读的16位递增和递增/递减计数寄存器P_TMRx_TCNT (x = 3, 4)
- 可读可写的16位定时周期寄存器：P_TMRx_TPR (x = 3, 4)，它提供了MCPx (x = 3, 4)的时间基准
- 4个可读可写的16位定时通用寄存器：P_TMRx_TGRA、P_TMRx_TGRB、P_TMRx_TGRC、P_TMRx_TGRD (x = 3, 4)，用于PWM的比较匹配输出
- 3个只读定时缓冲寄存器：P_TMRx_TBRA, P_TMRx_TBRB, P_TMRx_TBRC (x = 3, 4)，是以上提到的定时器的通用缓冲寄存器
- 可读可写的16位定时控制寄存器：P_TMRx_Ctrl和输入输出控制寄存器P_TMRx_IOCtrl (x = 3, 4)，用于模式的选择：PWM比较匹配输出模式
- 可读可写的16位定时器输出控制寄存器P_TMRx_OutputCtrl (x = 3, 4),决定了PWM的发生模式
- 可读可写的16位定时器中断使能寄存器P_TMRx_INT，提供两个中断源和一个16位只读中断状态寄存器P_TMRx_Status (x = 3, 4)，其中记录着各中断标志
- 可读可写的16位死区设置和控制寄存器P_TMRx_DeadTime (x = 3, 4)
- 可读可写的16位输入错误控制寄存器：P_Faultx_Ctrl (x = 1, 2)，错误释放寄存器P_Faultx_Release (x = 1, 2)和过载保护寄存器P_OLx_Ctrl (x = 1, 2)

表 13-1 MCP 定时器 3 和定时器 4 规格说明

功能	MCP定时器3	MCP定时器4

功能		MCP定时器3	MCP定时器4
时钟源		内部时钟：$F_{CK}/1$，$F_{CK}/4$，$F_{CK}/16$，$F_{CK}/64$，$F_{CK}/256$，$F_{CK}/1024$，外部时钟：TCLKA，TCLKB	内部时钟：$F_{CK}/1$，$F_{CK}/4$，$F_{CK}/16$，$F_{CK}/64$，$F_{CK}/256$，$F_{CK}/1024$ 外部时钟：TCLKA，TCLKB
输出引脚		TIO3A/U1，TIO3B/V1，TIO3C/W1，TIO3D/U1N,TIO3E/V1N，TIO3F/W1N	TIO4A/U2，TIO4B/V2，TIO4C/W2，TIO4D/U2N，TIO4E/V2N，TIO4F/W2N
定时通用寄存器		P_TMR3_TGRA，P_TMR3_TGRB，P_TMR3_TGRC，P_TMR3_TGRD	P_TMR4_TGRA，P_TMR4_TGRB，P_TMR4_TGRC，P_TMR4_TGRD
定时缓冲寄存器		P_TMR3_TBRA，P_TMR3_TBRB，P_TMR3_TBRC	P_TMR4_TBRA，P_TMR4_TBRB，P_TMR4_TBRC
中断周期		每1，2，4，8个周期发生中断	每1，2，4，8个周期发生中断
计数边沿		上升，下降，双沿计数	上升，下降，双沿计数
计数清除源		P_TMR3_TPR比较匹配后清除	P_TMR4_TPR比较匹配后清除
PWM比较匹配输出功能	1输出	Yes	Yes
	0输出	Yes	Yes
	输出保持	Yes	Yes
直流无刷电机PWM驱动		Yes	Yes
交流感应电机PWM驱动		Yes	Yes
边沿PWM		Yes	Yes
中心PWM		Yes	Yes
互补PWM		Yes	Yes
定时缓冲操作		是，但除了P_TMR3_TGRD	是，但除了P_TMR4_TGRD
AD转换启动触发器		P_TMR3_TGRD比较匹配	P_TMR4_TGRD比较匹配
PWM占空比同步载入		是，通过P_TMR_LDOK寄存器	是，通过P_TMR_LDOK寄存器
PWM输出使能控制		是，通过P_TMR_ Output寄存器	是，通过P_TMR_ Output寄存器
PWM波形控制	强制为高	Yes	Yes
	强制为低	Yes	Yes
	高有效	Yes	Yes
	低有效	Yes	Yes
UVW相位同步		与P_POS0_DectData寄存器变换同步,与P_TMR3_TGRB 或 P_TMR3_TGRC比较匹配同步	与P_POS1_DectData寄存器变换同步，与P_TMR4_TGRB 或 P_TMR4_TGRC比较匹配同步
占空模式		利用P_TMR3_TGRA寄存器或三个定时通用寄存器	利用P_TMR4_TGRA寄存器或三个定时通用寄存器
TPM寄存器写保护		是，通过P_TPWM_Write寄存器	是，通过P_TPWM_Write寄存器
外部错误输入引脚		FTIN1	FTIN2
外部过载输入引脚		OL1	OL2
中断源		定时器 3 TPR 中断 定时器 3 TGRD 中断 外部错误输入 1 中断 外部过载输入 1 中断 MCP 定时器 3 PWM 输出短路中断	定时器 4 TPR 中断 定时器 4 TGRD 中断 外部错误输入 2 中断 外部过载输入 2 中断 MCP 定时器 4 PWM 输出短路中断

13.3MCP 定时器 3 和定时器 4 输入/输出/特殊功能引脚

通道	引脚名称	I/O	功能
外部时钟输入	TCLKA	输入	外部时钟 A 输入引脚
	TCLKB	输入	外部时钟 B 输入引脚
MCP 定时器 3	TIO3A	输出	P_TMR3_TGRA PWM 比较匹配输出引脚
	TIO3B	输出	P_TMR3_TGRB PWM 比较匹配输出引脚
	TIO3C	输出	P_TMR3_TGRC PWM 比较匹配输出引脚
	TIO3D	输出	通过 P_TMR3_OutputCtrl 寄存器和 TIO3A 状态编程
	TIO3E	输出	通过 P_TMR3_OutputCtrl 寄存器和 TIO3B 状态编程
	TIO3F	输出	通过 P_TMR3_OutputCtrl 寄存器和 TIO3C 状态编程
	FTIN1	输入	外部错误输入引脚 1
	OL1	输入	外部过载输入引脚 1
MCP 定时器 4	TIO4A	输出	P_TMR4_TGRA PWM 比较匹配输出引脚
	TIO4B	输出	P_TMR4_TGRB PWM 比较匹配输出引脚
	TIO4C	输出	P_TMR4_TGRC PWM 比较匹配输出引脚
	TIO4D	输出	通过 P_TMR4_OutputCtrl 寄存器和 TIO4A 状态编程
	TIO4E	输出	通过 P_TMR4_OutputCtrl 寄存器和 TIO4B 状态编程
	TIO4F	输出	通过 P_TMR4_OutputCtrl 寄存器和 TIO4C 状态编程
	FTIN2	输入	外部错误输入引脚 2
	OL2	输入	外部过载输入引脚 2

13.4MCP 定时器的计数操作

SPMC75X 系列微控制器的 MCP 定时器有以下 5 种计数操作方式：

- 标准操作(标准递增计数)
- 根据外部时钟引脚 TCLKA 或 TCLKB 计数
- 边沿 PWM 模式(连续递增计数，PWM 输出模式)
- 中心 PWM 模式(递增/递减计数，PWM 输出模式)
- PWM 互补模式和死区时间控制

P_TMRx_Ctrl (x = 3, 4)寄存器中的 MODE 位的值决定了 MCP 定时器计数方式。当 P_TMR_Start 寄存器相应启动位置 1 后，MCP 定时器就从 0x0000 开始计数。P_TMRx_Ctrl (x = 3, 4)寄存器的 CCLS 位决定着计数器的清除源，P_TMRx_TPR (x = 3, 4)寄存器中的 PRDINT 位选择了中断的频率。

13.4.1 边沿 PWM 的连续递增计数模式

MCP 定时器的每个通道都可可由 P_TMRx_Ctrl (x = 3, 4)寄存器的 MODE 位设置为 PWM 输出模式或没有任何 PWM 输出的标准操作模式。在此模式下，定时器从 0x0000 开始递增计数，直到

与周期寄存器设置值相等为止。用户须配置 P_TMRx_TPR (x = 3 - 4)寄存器，将计数器清除源(CCLS)位设置成由定时器周期比较匹配清除，对 P_TMRx_Ctrl (x = 3, 4)的 PRDINT 选择恰当的值。

MCP 连续递增计数的频率来自输入的时钟源，该时钟源是由相应控制寄存器中 TMRPS 位区域选择。计数器连续计数，当计数值与周期定时寄存器的值相匹配时，计数器清零，周期比较匹配事件中断标志“TPRIF”置位，如果 P_TMRx_INT (x = 3, 4)寄存器中的 PPRIE 位为使能状态，则发生周期中断请求。

当定时计数器的内容与 P_TMRx_TGRD (x = 3, 4)比较匹配时，比较匹配事件发生，可以 AD 转换启动。

P_TMRx_TPR (x = 3, 4)的初始值可是 0x0000 到 0xFFFF 之间的任意值。定时器的时钟源可以选择外部时钟输入也可以选择内部时钟源 F_{CK}。标准连续递增计数模式非常适用于数字电机控制系统中的边沿触发或异步 PWM 以及周期采样。如图 13-2 所示为 MCP 定时器的标准递增计数模式。

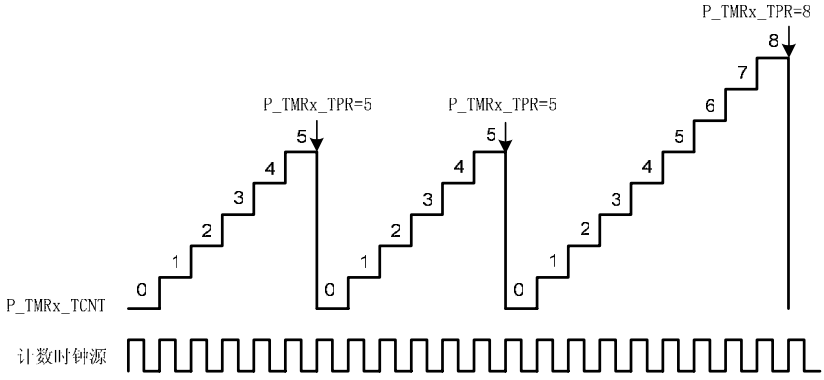


图 13-2 连续递增计数模式(CCLS = 111’b, CKEGS = 00’b, TMRPS = 000’b)

在边沿 PWM 模式中，用户必须设置 P_TMRx_TPR (x= 3, 4)周期寄存器和 P_TMRx_TGRy (y = A, B, C)通用寄存器的值，并且将计数清除源(CCLS)位设置成定时器周期清除。比较匹配输出模式在 P_TMRx_IOCtrl (x= 3, 4)寄存器中设置。如图 13-3 所示为 MCP 定时器的边沿 PWM 标准连续递增计数的模式。

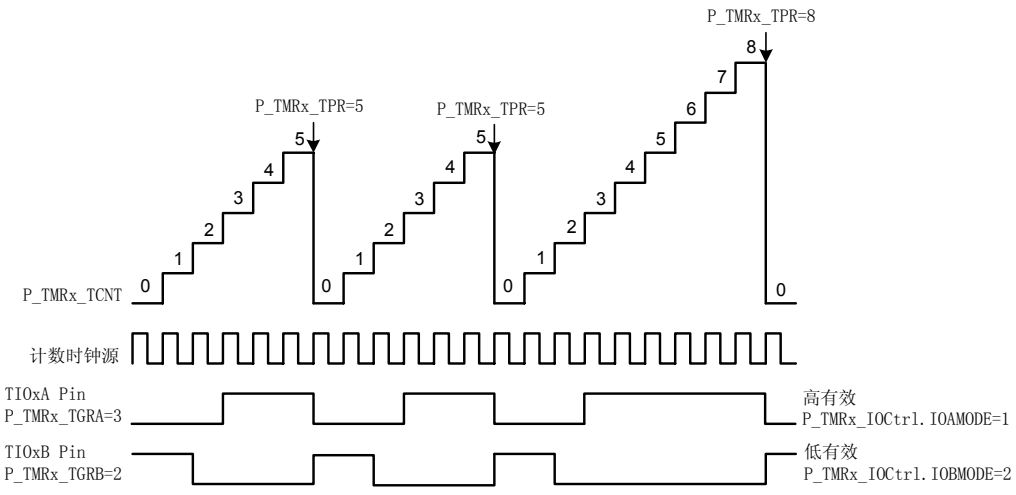


图 13-3 边沿 PWM 模式

13.4.2 中心 PWM 的连续递增/递减计数模式

连续递增/递减计数模式的操作与递增计数模式基本相同，唯一不同的是：定时器周期寄存器定义了整个计数过程的中间过渡点，计数过程中先递增，当达到定时周期寄存器设定值后开始递减。设置寄存器 P_TMRx_Ctrl (x = 3, 4)的 CKEGS 位，MCP 定时器的周期是寄存器 P_TMRx_TPR (x = 3, 4)设定值的两倍。如图 13-4 所示连续递增/递减计数模式的操作。

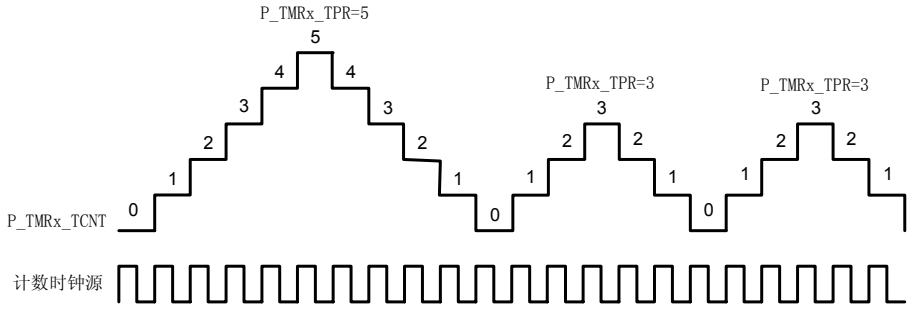


图 13-4 连续递增/递减计数模式(CCLS = 111'b, CKEGS = 00'b, TMRPS = 000'b)

定时器周期寄存器的初始值可以从 0x0000 到 0xFFFF 之间任意设定。当定时计数器的值与周期定时寄存器的值相等时，MCP 定时器开始向下计数直到 0。周期中断的方式与连续递增计数方式的描述相同。

P_TMRx_Status (x = 3, 4)寄存器中的 TCDF 位记录着计数的方向。定时器既可以选择外部时钟输入也可以选择内部时钟源 F_{CK}。如图 13-5 所示为定时器 3 中心 PWM 模式的递增/递减计数模式。

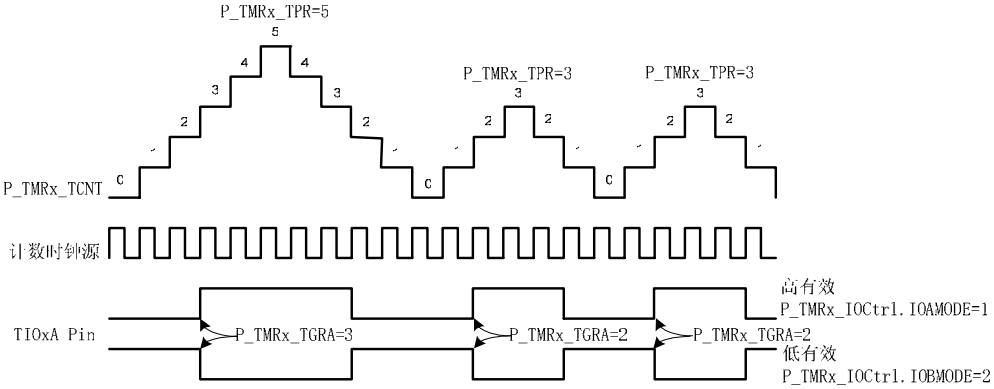


图 13-5 中心 PWM 模式

13.5MCP 定时器控制寄存器

SPMC75X 系列微控制器的 MCP 定时器 3 和 PDC 定时器 4 模块共有 40 个控制寄存器，如表 13-2 所示。通过这 40 个控制寄存器可以完成 MCP 定时器 3 和 MCP 定时器 4 模块所有功能的控制。

表 13-2 PDC 定时器 0 和 PDC 定时器 1 的控制寄存器

地址	寄存器	名称
7405h	P_TMR_Start	定时计数启动寄存器

地址	寄存器	名称
7406h	P_TMR_Output	定时输出使能寄存器
7409h	P_TPWM_Write	定时/PWM模块写使能控制寄存器
740Ah	P_TMR_LDOK	定时器载入完成寄存器
7403h	P_TMR3_Ctrl	定时器3控制寄存器
7413h	P_TMR3_IOCtrl	定时器3输入/输出控制寄存器
7423h	P_TMR3_INT	定时器3中断使能寄存器
7428h	P_TMR3_Status	定时器3中断状态寄存器
7407h	P_TMR3_OutputCtrl	定时器3输出控制寄存器
7460h	P_TMR3_DeadTime	定时器3死区控制寄存器
7466h	P_Fault1_Ctrl	错误输入1控制和状态寄存器
746Ah	P_Fault1_Release	错误1标志释放寄存器
7468h	P_OL1_Ctrl	过载输入1控制和状态寄存器
7433h	P_TMR3_TCNT	定时器3计数寄存器
7438h	P_TMR3_TPR	定时器3周期寄存器
7448h	P_TMR3_TGRA	定时器3通用寄存器A
7449h	P_TMR3_TGRB	定时器3通用寄存器B
744Ah	P_TMR3_TGRC	定时器3通用寄存器C
744Bh	P_TMR3_TGRD	定时器3通用寄存器D
7458h	P_TMR3_TBRA	定时器3缓冲寄存器A
7459h	P_TMR3_TBRB	定时器3缓冲寄存器B
745Ah	P_TMR3_TBRC	定时器 3 缓冲寄存器 C
7404h	P_TMR4_Ctrl	定时器4控制寄存器
7414h	P_TMR4_IOCtrl	定时器4输入/输出控制寄存器
7424h	P_TMR4_INT	定时器4中断使能寄存器
7429h	P_TMR4_Status	定时器4中断状态寄存器
7408h	P_TMR4_OutputCtrl	定时器4输出控制寄存器
7461h	P_TMR4_DeadTime	定时器4死区控制寄存器
7467h	P_Fault2_Ctrl	错误输入2控制和状态寄存器
746Bh	P_Fault2_Release	错误2标志释放寄存器
7469h	P_OL2_Ctrl	过载输入2控制和状态寄存器
7434h	P_TMR4_TCNT	定时器4计数寄存器
7439h	P_TMR4_TPR	定时器4周期寄存器
744Ch	P_TMR4_TGRA	定时器4通用寄存器A
744Dh	P_TMR4_TGRB	定时器4通用寄存器B
744Eh	P_TMR4_TGRC	定时器4通用寄存器C
744Fh	P_TMR4_TGRD	定时器4通用寄存器D
745Ch	P_TMR4_TBRA	定时器4缓冲寄存器A
745Dh	P_TMR4_TBRB	定时器4缓冲寄存器B
745Eh	P_TMR4_TBRC	定时器 4 缓冲寄存器 C

13.5.1 MCP 定时器的控制寄存器

P_TMRx_Ctrl (x = 3, 4)配置了时钟源的选择，计数时钟边沿，计数器清除源，TPR 中断频率和定时器操作模式。TCLKA、TCLKB 时钟输入是由系统时钟 F_{CK} 采样获得的，并忽略脉冲宽度小于采样时钟 4 倍的脉冲。MCP 定时器 3 和定时器 4 不支持输入捕获。当定时器处于双沿编程状态时，计数时钟是输入的时钟 2 倍频。

P_TMR3_Ctrl ($7403)：MCP 定时器 3 控制寄存器

P_TMR4_Ctrl ($7404)：MCP 定时器 4 控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
PRDINT		MODE				保留	

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CCLS			CKEGS		TMRPS		

第 15:14 位　**PRDINT**：TPR 中断频率选择。

00 ＝ 每个周期中断一次

01 ＝每 2 个周期中断一次

10 ＝每 4 个周期中断一次

11 ＝每 8 个周期中断一次

第 13:10 位　**MODE**：模式选择，用于选择 MCP 定时器操作模式

0xxx ＝ 标准操作(连续递增计数)

1x0x ＝ 边沿 PWM 模式 (连续递增计数, PWM 输出)

1x1x ＝ 中心 PWM 模式(递增/递减计数, PWM 输出)

第 9:8 位　保留

第 7:5 位　**CCLS**：定时清除源选择。用于选择 TCNT (x = 3, 4)计数清除源

000 ＝ TCNT 清除禁止

001 ＝ 保留

010 ＝ 保留

011 ＝ 保留

100 ＝ 保留

101 ＝ 保留

110 ＝ 保留

111 ＝ TCNT 与 P_TMRx_TPR (x = 3, 4)比较匹配时清除

第 4:3 位　**CKEGS**：时钟边沿选择，这些位用于选择时钟输入的边沿。当输入时钟为

双沿时，输入时钟周期就减半。当选择 FCK/1 作为计数时钟时，如果在选择双沿触发条件下，那么计数器将只在上升沿计数。

00 ＝ 上升沿计数

01 ＝ 下降沿计数

1X ＝ 双沿计数

第 2:0 位　**TMRPS**：定时器分频选择。选择 TCNT 计数器的时钟源。每个定时器通道可独立选择各自的时钟源。

000 ＝ F_{CK}/1 计数

001 ＝ F_{CK}/4 计数

010 ＝ F_{CK}/16 计数

011 ＝ F_{CK} /64 计数

100 ＝ F_{CK} /256 计数

101 ＝ F_{CK}/1024 计数

110 ＝ 由 TCLKA 引脚输入

111 ＝ 由 TCLKB 引脚输入

13.5.2 MCP 定时器的输出控制寄存器

P_TMRx_IOCtrl (x =3, 4)寄存器控制着 TIOxA、TIOxB 和 TIOxC (x = 3, 4)引脚上 PWM 比较匹配输出的操作方式。通过设置 P_TMRx_Ctrl (x = 3, 4)寄存器中 CCLS 和 MODE 位的值可以确定定时器操作模式。当选择 PWM 输出模式时，IOAMODE/IOBMODE/IOCMODE 这些位决定着波形输出模式。MCP 定时器 3 和 MCP 定时器 4 没有输入捕获操作功能，因此，IOAMODE/IOBMODE/IOCMODE 的值 1xxx'b 为无效。

P_TMR3_IOCtrl ($7413)：MCP 定时器 3 输出控制寄存器

P_TMR4_IOCtrl ($7414)：MCP 定时器 4 输出控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
保留				IOCMODE			

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
IOBMODE				IOAMODE			

第 15:12 位　保留

第 11:8 位　**IOCMODE**：选择定时器 3/定时器 4 IOC 的设置

第 7:4 位　**IOBMODE**：选择定时器 3/定时器 4 IOB 的设置

第 3:0 位: **IOAMODE**：选择定时器 3/定时器 4 IOA 的设置

PWM 比较匹配输出模式:

0000 = 初始输出为 0, 比较匹配时输出 0

0001 = 初始输出为 0, 比较匹配时输出 1

0010 = 初始输出为 1 比较匹配时输出 0

0011 = 初始输出为 1, 比较匹配时输出 1

01xx = 输出保持

1xxx = 保留

13.5.3 MCP 定时器的中断使能寄存器

P_TMRx_INT (x = 3, 4)寄存器用来使能或禁止由 TGRD 比较匹配发出的 A/D 转换启动请求的，中断请求包括周期寄存器比较匹配和 TGRD 寄存器比较匹配。

P_TMR3_INT ($7423)：MCP 定时器 3 中断使能寄存器

P_TMR4_INT ($7424)：MCP 定时器 4 中断使能寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R	R	R/W	R/W	R	R	R
0	0	0	0	0	0	0	0
TADSE	保留		TPRIE	TGDIE	保留		

第 15:8 位 保留

第 7 位 **TADSE**：A/D 转换启动请求使能位。通过 TGRD 寄存器的比较匹配结果使能或禁止 A/D 转换启动请求

0 = 禁止

1 = 使能

第 6:5 位 保留

第 4 位 **TPRIE**：定时器周期寄存器中断使能位。使能或禁止 TPR 寄存器比较匹配的中断请求。

0 = 禁止

1 = 使能

第 3 位 **TGDIE**：定时通用寄存器 D 中断使能位。使能或禁止 TGRD 寄存器比较匹配中断请求。

0 = 禁止

1 = 使能

第 2:0 位　　保留

13.5.4 MCP 定时器的中断状态寄存器

中断状态寄存器中指示了周期寄存器比较匹配和 TGRD 比较匹配事件的发生情况。这些标志位指明了中断源，当 P_TMRx_INT (x = 3, 4)寄存器里相应中断使能位被置 1 后就会发生中断。

当定时器设置在中心 PWM 模式下，TCDF 位表示计数器当前的计数方向。

P_TMR3_Status ($7428)：MCP 定时器 3 的中断状态寄存器

P_TMR4_Status ($7429)：MCP 定时器 4 的中断状态寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R	R	R/W	R/W	R	R	R
0	0	0	0	0	0	0	0
TCDF	保留		TPRIF	TGDIF	保留		

第 15:8 位　　保留

第 7 位　　TCDF：计数方向标志。用于表示中心 PWM 模式下 TCNT 计数的方向

0 = 递增计数

1 = 递减计数

第 6:5 位　　保留

第 4 位　　TPRIF：定时器周期寄存器比较匹配标志。用于表示 TPR 寄存器比较匹配事件是否发生。

写入“1”清除该标志。

0 = 比较匹配事件没有发生

1 = 比较匹配事件已发生

第 3 位　　TGDIF：定时通用寄存器 D 比较匹配标志。该标志用于指示一个 TGRD 寄存器比较匹配输出事件是否已经发生。写入“1”清除该标志。

0 =比较匹配事件没有发生

1 =比较匹配事件已发生

第 2:0 位　　保留

13.5.5 MCP 定时器的输出使能寄存器

该寄存器使能/禁止 MCP3 和 MCP4 的 PWM 输出功能，若 PWM 输出功能禁止，PWM 输出将置为高阻态。注意，该寄存器只有当 TIO3A-TIO3F 或者 TIO4A-TIO4F 在特殊功能模式中被配置为输出引脚时候才能生效。

P_TMR_Output ($7406)：MCP 定时器输出使能寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
保留		TMR4FOE	TMR4EOE	TMR4DOE	TMR4COE	TMR4BOE	TMR4AOE

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
保留		TMR3FOE	TMR3EOE	TMR3DOE	TMR3COE	TMR3BOE	TMR3AOE

第 15:14 位　　保留

第 13 位　　**TMR4FOE：** MCP 定时器 4 IOF 输出使能(TIO4F)。
该位使能/禁止了 TIO4F 引脚输出。
0 = 禁止
1 = 使能

第 12 位　　**TMR4EOE：** MCP 定时器 4 IOE 输出使能(TIO4E)。
该位使能/禁止了 TIO4E 引脚输出。
0 = 禁止
1 = 使能

第 11 位　　**TMR4DOE：** MCP 定时器 4 IOD 输出使能(TIO4D)。
该位使能/禁止了 TIO4D 引脚输出。
0 = 禁止
1 = 使能

第 10 位　　**TMR4COE：** MCP 定时器 4 IOC 输出使能 (TIO4C)。
该位使能/禁止了 TIO4C 引脚输出。
0 = 禁止
1 = 使能

第 9 位　　**TMR4BOE：** MCP 定时器 4 IOB 输出使能 (TIO4B)。
该位使能/禁止了 TIO4B 引脚输出。
0 = 禁止
1 = 使能

第 8 位　　**TMR4AOE：** MCP 定时器 4 IOA 输出使能 (TIO4A)。

该位使能/禁止了 TIO4A 引脚输出。

0 = 禁止

1 = 使能

第 7:6 位 保留

第 5 位 **TMR3FOE：** MCP 定时器 3 IOF 输出使能(TIO3F)。

该位使能/禁止了 TIO3F 引脚输出。

0 = 禁止

1 = 使能

第 4 位 **TMR3EOE：** MCP 定时器 3 IOE 输出使能(TIO3E)。

该位使能/禁止了 TIO3E 引脚输出。

0 = 禁止

1 = 使能

第 3 位 **TMR3DOE：** MCP 定时器 3 IOD 输出使能(TIO3D)。

该位使能/禁止了 TIO3D 引脚输出。

0 = 禁止

1: Enable

第 2 位 **TMR3COE：** MCP 定时器 3IOC 输出使能(TIO3C)。

该位使能/禁止了 TIO3C 引脚输出。

0 = 禁止

1 = 使能

第 1 位 **TMR3BOE：** MCP 定时器 3IOB 输出使能(TIO3B)。

该位使能/禁止了 TIO3B 引脚输出。

0 = 禁止

1 = 使能

第 0 位 **TMR3AOE：** MCP 定时器 3IOA 输出使能(TIO3A)。

该位使能/禁止了 TIO3A 引脚输出。

0 = 禁止

1 = 使能

13.5.6 MCP 定时器的输出控制寄存器

MCP 定时器 3 和 MCP 定时器 4 输出控制寄存器的设置对于电机驱动应用中 PWM 波形类型是非常重要的。“DUTYMODE”位决定了 PWM 占空比模式，一般而言，在 120 度 PWM 模式下，采用共用有 P_TMRx_TGRA (x = 3, 4)模式既只需要设置占空比的值，在 180 度 PWM 模式下 BLDC 或 ACI 下，对三个寄存器 P_TMRx_TGRA/P_TMRx_TGRB/P_TMRx_TGRC (x = 3, 4)都需要设置。图 13-16 显示了（DUTYMODE = 0，1） PWM 输出的时序。图 13-17 显示了输出极性变化的 PWM 输出时序：

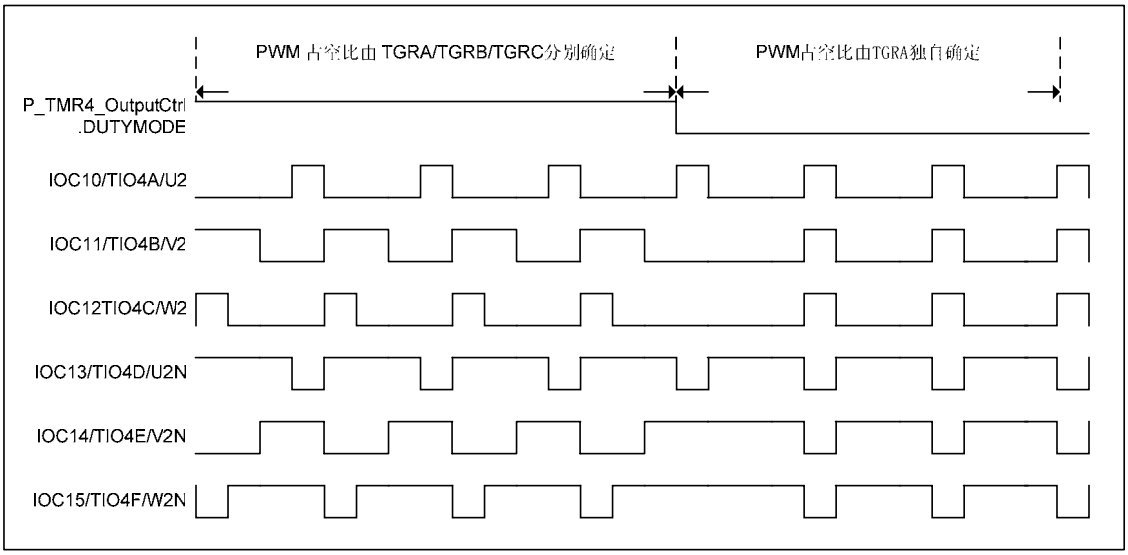


图 13-1　DUTYMODE 不同的 PWM 输出时序

POLP 位设定了 PWM 对 IGBT/MOSFET 开关的有效电平。可以通过设置 UPWM, VPWM 和 WPWM 强制 PWM 输出为高/低电平或在特定引脚以波形方式输出。POLP, WPWM/VWPM/UPWM 和 WOC/VOC/UOC 这些位决定了 PWM 波形发生的种类。表 13-2 和 13-3 列出了 U 相输出极性和波形控制，此方式同样适用于 V 相和 W 相的描述。图 13-7 显示 PWM 的极性.

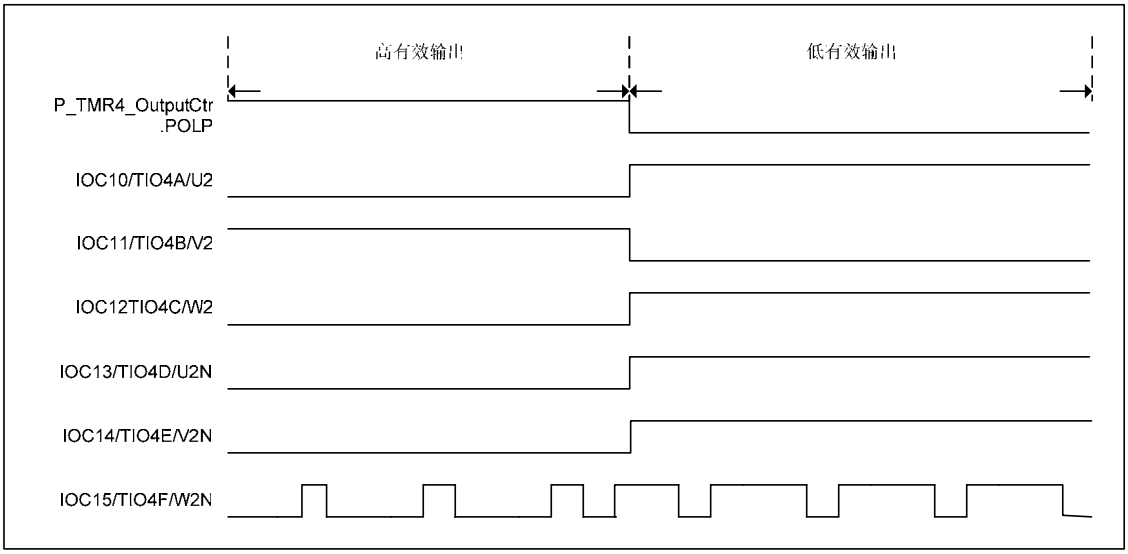


图 13-7 输出极性变化的 PWM 输出时序

P_TMR3_OutputCtrl ($7407)：MCP 定时器 3 输出控制寄存器

P_TMR4_OutputCtrl ($7408)：MCP 定时器 4 输出控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	RW	R	R	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0

DUTYMODE	POLP	保留	WPWM	VPWM	UPWM

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SYNC		WOC		VOC		UOC	

第 15 位　DUTYMODE：占空模式选择。此位对补偿 PWM 输出占空模式进行选择。

0 = 共用 TGRA 寄存器

1 = 三相独立

第 14 位　POLP：上相极性选择。用于选择上相的极性

0 = 低有效

1 = 高有效

第 13:11 位　保留

第 10 位　WPWM：W 相 PWM 输出选择

0 = H/L 电平输出

1 = PWM 波形输出

第 9 位　VPWM：V 相 PWM 输出选择

0 = H/L 电平输出

1 = PWM 波形输出

第 8 位　UPWM：U 相 PWM 输出选择

0 = H/L 电平输出

1 = PWM 波形输出

第 7:6 位　SYNC：UVW 相位输出同步源选择

00 = 不同步

01 = 与 P_POSx_DectData (x = 0, 1) 寄存器变化同步

10 = 与 TGRB 寄存器比较匹配同步

11 = 与 TGRC 寄存器比较匹配同步

第 5:4 位　WOC：W 相输出控制

第 3:2 位:　VOC：V 相输出控制

第 1:0 位　UOC：U 相输出控制

表 13-2 和 13-3 列出了根据在 P_TMRx_OutputCtrl (x = 3, 4) 寄存器中 POLP 和 UOC/VOC/WOC 这些位的设置所选择的 PWM 输出波形种类。

在 P_TMRx_OutputCtrl (x = 3, 4)寄存器中定义了四种 PWM 波形输出模式：模式 0-模式 3。通过对[10:8]位和[7:0]位的设置，可以在 Ux/UxN, Vx/VxN 和 Wx/WxN (x= 1, 2)输出的每对引脚上产生不同的 PWM 波形。注意 POLP 位和 WPWM、VPWM、UPWM 位决定着波形的有效性是根据 P_TMRx_TCNT (x = 3, 4)计数时钟源或者仅仅是强制逻辑高输出或者低跳变。

模式 0 和模式 3 用于产生互补的 PWM 波形。典型用法是产生正弦 PWM 波形驱动交流感应电机。这些模式也可以应用于直流无刷电机的 180 度变频驱动控制方式。在模式 0 和模式 3 下产生互补的 PWM 波形时，DTP 位设置死区控制来保护驱动电路，DTWE、DTVE、DTUE 位应当设置成 1，从而可以使开发者实现指定相位上的死区控制使能。

模式 1 和模式 2 是标准的 PWM 模式，相关相位的死区特性是禁止的。

表 13-2 U 相输出极性：高有效(POLP=1)

UOC[1,0]			UPWM			
			1: PWM output		0: H/L output	
			U phase	UN phase	U phase	UN phase
Mode 0	0	0	CPWM	PWM	L	L
Mode 1	0	1	L	PWM	L	H
Mode 2	1	0	PWM	L	H	L
Mode 3	1	1	PWM	CPWM	H	H

表 13-3 U 相输出极性：低有效(POLP=0)

UOC[1,0]			UPWM			
			1: PWM output		0: H/L output	
			U phase	UN phase	U phase	UN phase
Mode0	0	0	PWM	CPWM	H	H
Mode1	0	1	H	CPWM	H	L
Mode2	1	0	CPWM	H	L	H
Mode3	1	1	CPWM	PWM	L	L

SYNC 决定了 U V W PWM 相位由内部事件来同步，与 P_POSx_DectData (x = 0, 1) 寄存器变化同步，与 TGRB 寄存器比较匹配同步，与 TGRC 寄存器比较匹配同步，图 13-8 描述 MCP 定时器 4 的 PWM 输出的同步模式。

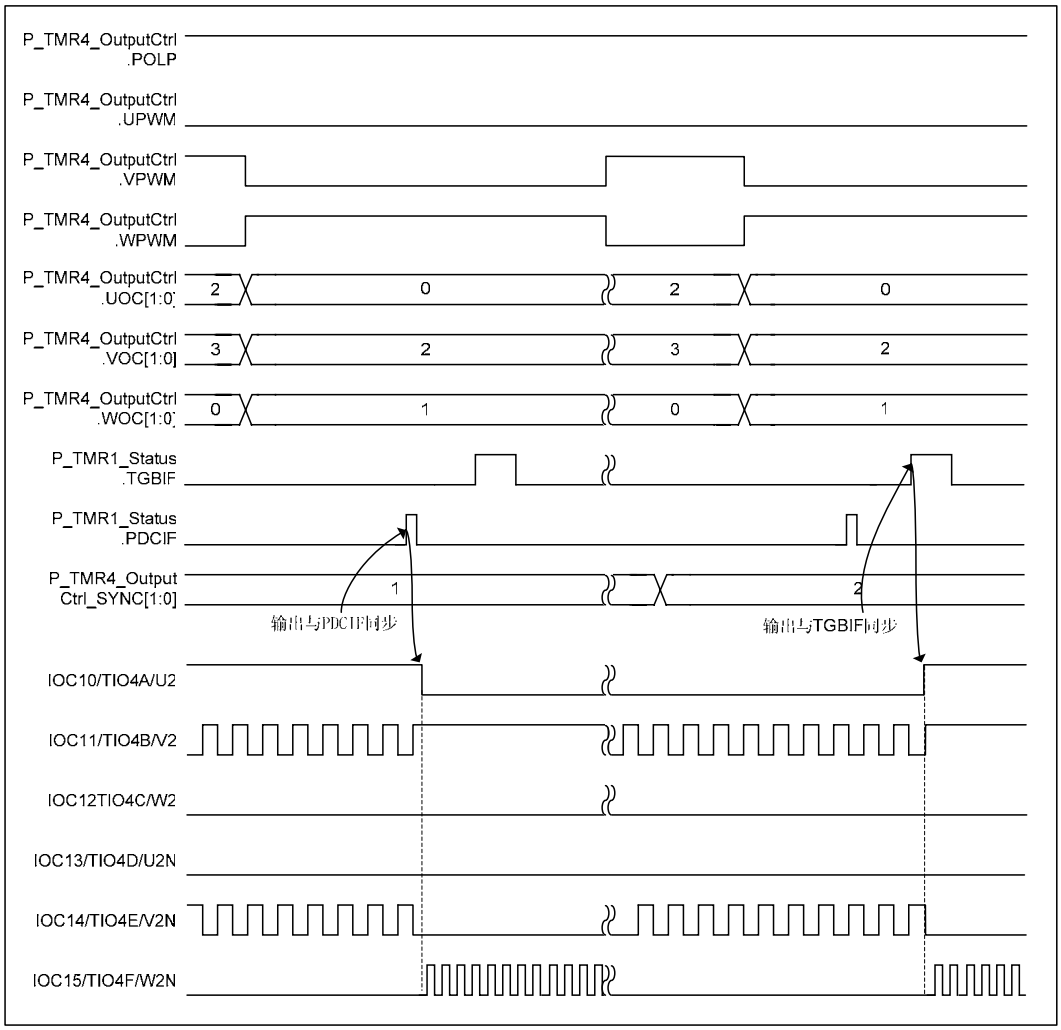


图 13-8　PWM 输出的同步模式

13.5.7　MCP 定时器的死区及其控制寄存器

在补偿 PWM 模式下，每对 PWM 补偿通道都可以用来驱动功率管的高端和低端。每对 PWM 信号应该在理想情况下逻辑是完全相反的，但实际情况并非如此。为了避免 PWM 在有效时间内高/低重叠，PWM 补偿模式中必须插入死区时间单元。如图 13-9 所示为 MCP 定时器 3 的中心补偿 PWM 模式下插入了死区时间的例子。

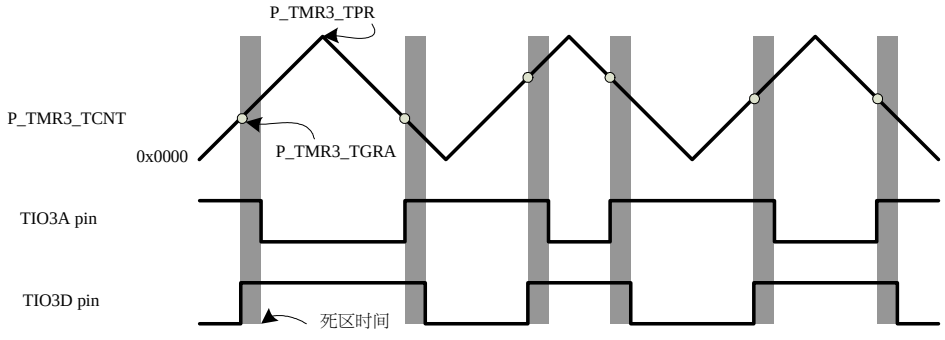


图 13-9 低有效 PWM 模式的死区发生

SPMC75X 系列微控制器有两个死区定时控制寄存器：P_TMR3_DeadTime 和 P_TMR4_DeadTime 分别服务于 MCP 定时器 3 和 MCP 定时器 4，只有在补偿 PWM 模式下死区时间定时器才可以工作。依据 P_TMRx_OutputCtrl (x = 3, 4)寄存器的 POLP 位的设置，死区定时单元会将高/低相位延迟输出。三相死区控制可以独立的使能或禁止，死区时间通过 DTP 位来设定，大小是 DTP*（F_{CK}/4 的周期）。

P_TMR3_DeadTime ($7460)：MCP 定时器 3 死区及其控制寄存器

P_TMR4_DeadTime ($7461)：MCP 定时器 4 死区及其控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R/W	RW	R/W	R	R	R	R
0	0	0	0	0	0	0	0
保留	DTWE	DTVE	DTUE	保留			

B7	B6	B5	B4	B3	B2	B1	B0
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
保留	DTP						

第 15 位　保留

第 14 位　DTWE：W 相的死区时间定时使能

0 = 禁止

1 = 使能

第 13 位　DTVE：V 相的死区时间定时使能

0 = 禁止

1 = 使能

第 12 位　DTUE：U 相的死区时间定时使能

0 = 禁止

1 = 使能

第 11:7 位　保留

第 6:0 位　DTP：死区定时周期。这些位选择了死区的周期，死区时间可以在 0 到 127 F_{CK}/4 时钟周期之间设置。

13.5.8 MCP 定时器的错误输入控制寄存器

错误保护输入是检测 FTINT1-2 引脚上出现低有效电平,而使输 PWM 出处于高阻态，起到保护作用，而且同时将会产生中断，见表 13-4。PWM 输出将会保持在高阻状态下直到解除该状态。图 13-10 显示了 IOC9/FTIN2 错误保护输入时序。

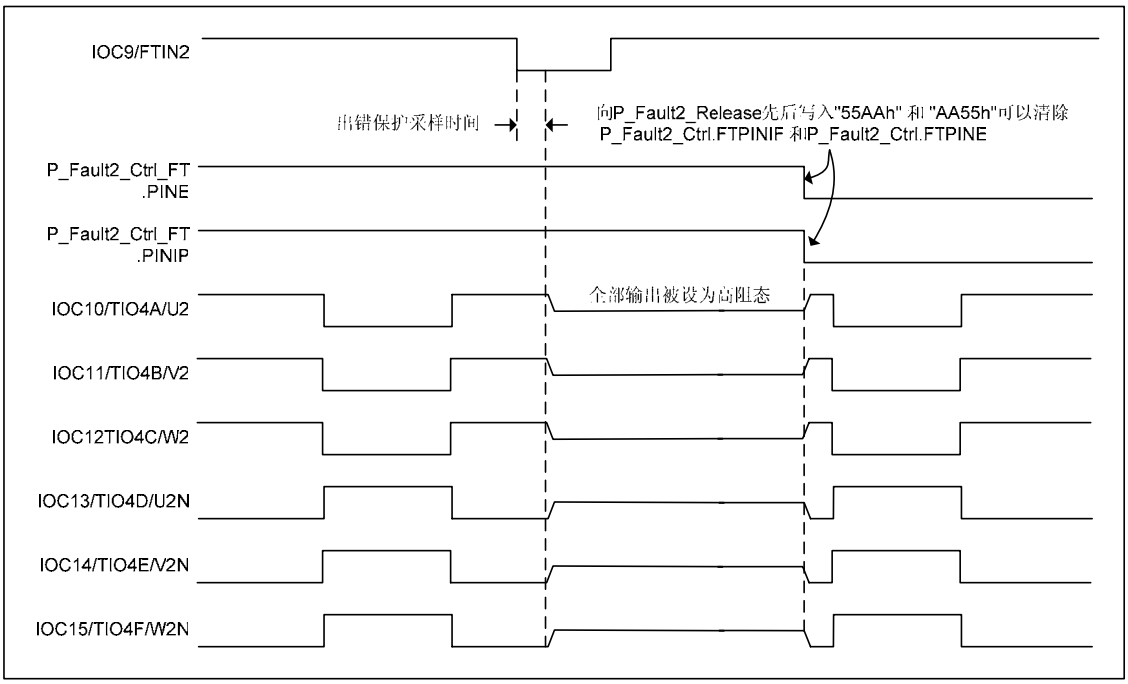


图 13-10 错误输入时序

还需要注意的是 P_Faultx_Ctrl (x = 1, 2)寄存器的 OCLS 位决定 PWM 输出比较的电平极性；用户需正确的选择 PWM 保护电平，保证目标系统驱动电路的安全。注：错误输入保护只在补偿 PWM 模式下才有效。图 13-11 显示了输出比较错误保护时序。

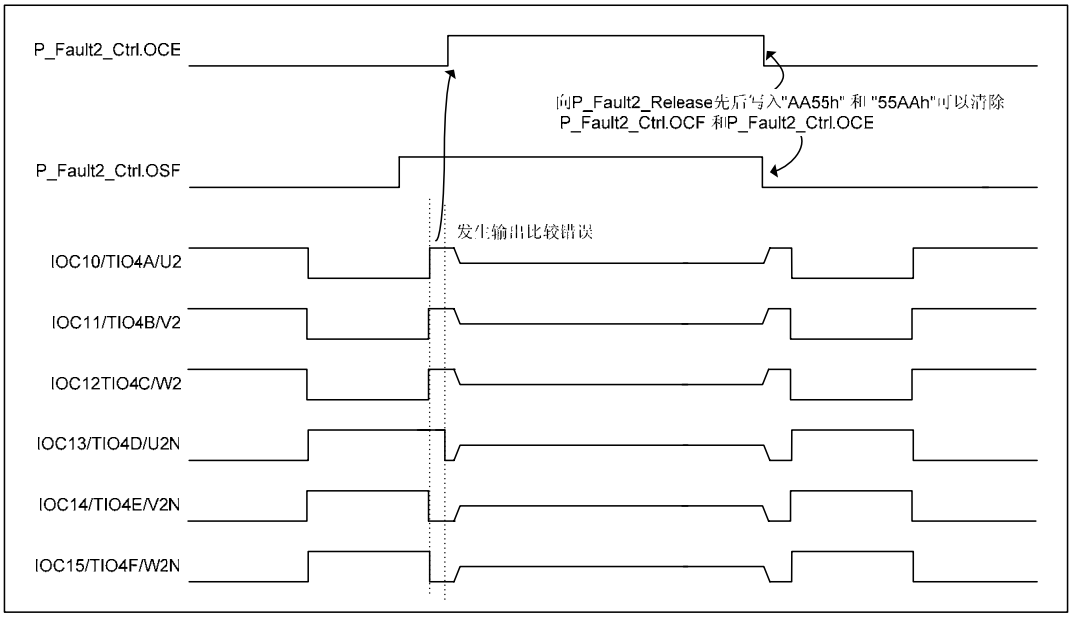


图 13-11 输出比较错误保护时序

MCP 定时器模块只能工作在稳定的时钟下，一旦检测到时钟出错，PWM 输出呈高阻态状态 ，保证目标系统驱动电路的安全。图 13-12 显示振荡器停止（保护）时序.

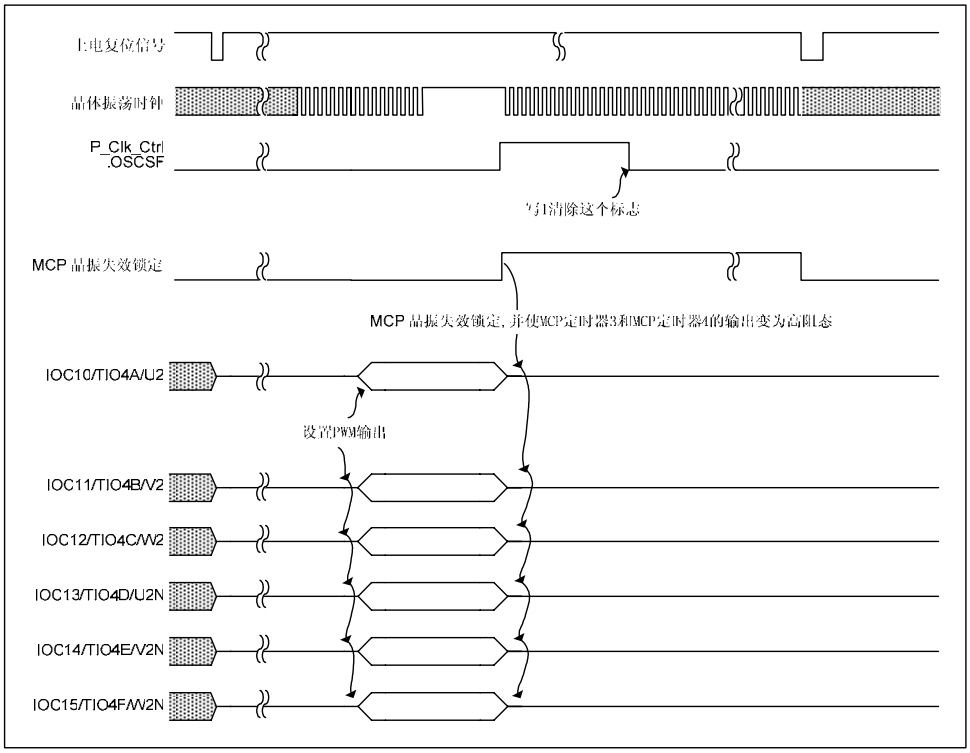


图 13-12 振荡器停振保护时序

OSF 和 FTPINIF 位只能由上电复位来初始化其值，系统复位不能初始化其值。错误保护状态(PWM 高阻态)只能靠上电复位或软件释放该状态，外部复位引脚的复位不能释放错误保护位的状态。

表 13-4 错误输入和 PWM 输出引脚

引脚名称	引脚状态	描述
FTIN1	输入	输入请求将 U1, V1, W1, U1N, V1N, W1N 设置成高阻态
FTIN2	输入	输入请求将 U2, V2, W2, U2N, V2N, W2N 设置成高阻态
U1, U1N	输入 悬浮	如果这两个引脚的输出同时为低电平至少一个周期，所有的 MCP3 的 PWM 输出引脚都将被设置成高阻态。
V1, V1N	输入 悬浮	
W1, W1N	输入 悬浮	
U2, U2N	输入 悬浮	如果这两个引脚的输出同时为低电平至少一个周期，所有的 MCP4 的 PWM 输出引脚都将被设置成高阻态。
V2, V2N	输入 悬浮	
W2, W2N	输入 悬浮	

PWM 输出在以下情况下将会停止(置为高阻态)

- 向错误输入引脚输入“低”(FTIN1/FTIN2)
- 上下相的输出电平相等
- 锁相环或晶振停止工作
- PWM 输出引脚可通过 FTIN1-2 引脚下降沿或低电平采样设置成高阻态
- 有效的 FTINT 输入可设置成在 $F_{CK}/4 \times 16$, $F_{CK}/16 \times 16$, $F_{CK}/64 \times 16$, 或 $F_{CK}/256 \times 16$

下的低电平保持

- 如果上下位同时输出了有效电平超过一个系统时钟周期，PWM 补偿输出可以设置成高阻态。
- 如果上 P_INT_Priority 寄存器 FTIP 位设置 ，就会产生快速中断请求。

P_Fault1_Ctrl ($7466)：MCP 错误输入 1 控制和状态寄存器

P_Fault2_Ctrl ($7467)：MCP 错误输入 2 控制和状态寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	RW	R/W	R/W	R	R	R	R
0	0	0	0	0	0	0	0
OCE	OCIE	OCLS	OSF	保留			

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
FTPINE	FTPINIE	FTPINIF	保留	FTCNT			

第 15 位: OCE：输出比较使能。使能/禁止输出电平比较。如果输出电平比较功能使能，将会比较三对 PWM 输出管脚，如果上下位同时输出了有效电平(输出短路)大于 一个时钟周期，所有的 PWM 输出将会置为高阻态。向 P_Faultx_Release (x = 1, 2)寄存器先写入”AA55h”再写入“55AAh”才会解除输出比较保护状态(PWM 输出高阻态)并使能 PWM 输出功能。同时 OSF 标志会被清除。该位在禁止时读出值为 0。

0 = 禁止

1 = 使能

第 14 位: OCIE：输出比较中断使能。该位决定在已经侦测到 PWM 输出短路时是否发出中断请求。

0 = 禁止

1 = 使能

第 13 位 OCLS：输出比较电平极性选择。该位选择了输出比较电平的极性。

0 = 比较低电平

1 = 比较高电平

第 12 位 OSF：输出短路标志，该标志指示了 PWM 输出短路是否已经被侦测到。向 P_Faultx_Release (x = 1, 2)写入“AA55h”再写入“55AAh”将清除该标志并且禁止了错误输入引脚。重新使能该引脚的错误保护功能需要软件将 OCE 再一次置 1。

第 11:8 位 保留

第 7 位 FTPINE：错误输入引脚 1/2 使能。该位使能/禁止了 FTINP1/2 引脚的输入。向 P_Faultx_Release (x = 1, 2)写入“55AAh” 后再写

入“AA55h”将会解除错误保护状态(PWM 输出高阻态)，并禁止了 FTINP1/2 引脚的输入，同时 FTPINIF 标志也会被清除，重新使能该引脚的错误保护功能需要软件将 FTPINE 再一次置 1。该位在禁止时读出值为 0。

0 = 禁止

1 = 使能

第 6 位 **FTPINIE**：错误输入 1/2 中断使能。该位使能/禁止 TINP1/2 引脚中断。

0 = 禁止

1 = 使能

第 5 位 **FTPINIF**：错误输入 1/2 状态标志。该位指示在 FTINP1/2 上是否发生了高阻态请求。向 FTRR1/2 写入“55AAh” 后再写入“AA55h”将会清除该标志。重新使能错保护功能需要软件将 FTPINE 位再一次置 1。

0 = 未发生

1 = 已发生

第 4 位 **保留**

第 3:0 位 **FTCNT**：错误保护采样间设置，采样时间为 $F_{CK}/4 * n$（n = 1 到 15）。注意： FTCNT 不能设置为 0。如果将 FTCNT 设置为 0，错误保护电路将认为出错信号一直发生，即使 FTIN1/2 引脚保持在高状态。如果这时 FTPINIE 设置为 1，FTPINIF 标志将不能被清除，错误保护中断程序将一直执行，这将使系统进入不可预知的状态。

13.5.9 MCP 定时器的错误释放寄存器

为了释放由错误输入引起的 PWM 输出高阻状态，首先断定 P_Faultx_Ctrl (x = 1, 2)寄存器中的 FTPINIF 位是否置位，然后向相应的错误解除寄存器 P_Faultx_Release (x = 1, 2) 连续写入“55AAh”和“AA55h”。

为了将 PWM 从输出短路造成的高阻态中释放，首先要检查 P_Faultx_Ctrl (x = 1, 2)寄存器中的 OSF 短路标志是否置位，然后向相应的错误解除寄存器 P_Faultx_Release (x = 1, 2)连续写入“AA55h” 和“55AAh”。

为了将 PWM 从晶振失效造成的高阻态中解除，首先要清除 P_Clk_Ctrl 寄存器中的振荡器失效标志 OSCSF，然后向相应的错误解除寄存器 P_Faultx_Release (x = 1, 2)连续写入“5555h” and “AAAAh”。

P_Fault1_Release($746A)：MCP 错误 1 标志解除寄存器

P_Fault2_Release($746B)：MCP 错误 2 标志解除寄存器

B15	B14	B13	B12	B11	B10	B9	B8

W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
FTRR							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
FTRR							

第 15:0 位　　**FTRR**: 错误解除控制字

13.5.10　MCP 定时器的过载保护控制和状态寄存器

SPMC75X 系列微控制器芯片包含有过载保护电路。当过载保护输入引脚(OL)拉低时，该电路开始工作。过载保护输入采样时钟为 $F_{CK}/4$，采样个数可以从 1 到 15。

有三种方法可以解除过载保护：由定时器释放、延迟 1 个周期释放或自动释放。当过载保护输入引脚(OL)已经恢复高电平时可以使用以上三种方法释放。

发生过载时保护方式可以设置为：无禁止（相位正常输出）、禁止所有相位输出、禁止 PWM 相位输出或禁止上/下相输出。禁止所有相位输出、禁止 PWM 相位输出适用于方波驱动马达时的过载保护，禁止上/下相输出适用于正弦波驱动马达时的过载保护。要禁止某相意味着将此相信号置于无效的电平上。

当选择为禁止上/下相输出（P_OLx_Ctrl.OLMD = 3, x = 1, 2）时，驱动马达的 PWM 输出状态由禁止前的开启瞬时状态决定。当两个或更多的上相有效时，所有的上相都打开并且所有的下相都关闭；当两个或更多的下相有效时，所有的下相都打开并且所有的上相都关闭。

表 13-5 和表 13-6 描述了 POLP 位（在 P_TMRx_OutputCtrl (x = 3, 4)寄存器中）和 OLMD 位的不同设置对应的发生过载保护的不同状态。图 13-13 显示了过载 PWM 停止时序。

表 13-5　过载保护状态（POLP = 1）

OLMD		TIOxA – TIOxF 相位输出状态 (x = 3, 4)	过载保护中断发生状态
0	0	无禁止（相位正常输出）	不发生中断
0	1	禁止所有相位输出	发生中断
1	0	禁止 PWM 相位输出	发生中断
1	1	(1) 当检测到上相的任意两相为高电平则禁止所有下相 PWM。 (2) 当检测到下相的任意两相为高电平则禁止所有上相 PWM。 (3) 当发生过载时，但上相的任意两相或下相的任意两相不是同时为高电平，则不会禁止相位，PWM 正常输出。	发生过载，同时检测到上相的任意两相或下相的任意两相为高电平则发生中断，否则不发生中断

表 13-6　过载保护状态（POLP = 0）

OLMD		TIOxA – TIOxF 相位输出状态 （x = 3, 4）	过载保护中断发生状态
0	0	无禁止（相位正常输出）	不发生中断
0	1	禁止所有相位输出	发生中断
1	0	禁止 PWM 相位输出	发生中断
1	1	(4) 当检测到上相的任意两相为低电平则禁止所有下相 PWM。 (5) 当检测到下相的任意两相为低电平则禁止所有上相 PWM。 (6) 当发生过载时，但上相的任意两相或下相的任意两相不是同时为低电平，则不会禁止相位，PWM 正常输出。	发生过载，同时检测到上相的任意两相或下相的任意两相为低电平则发生中断，否则不发生中断

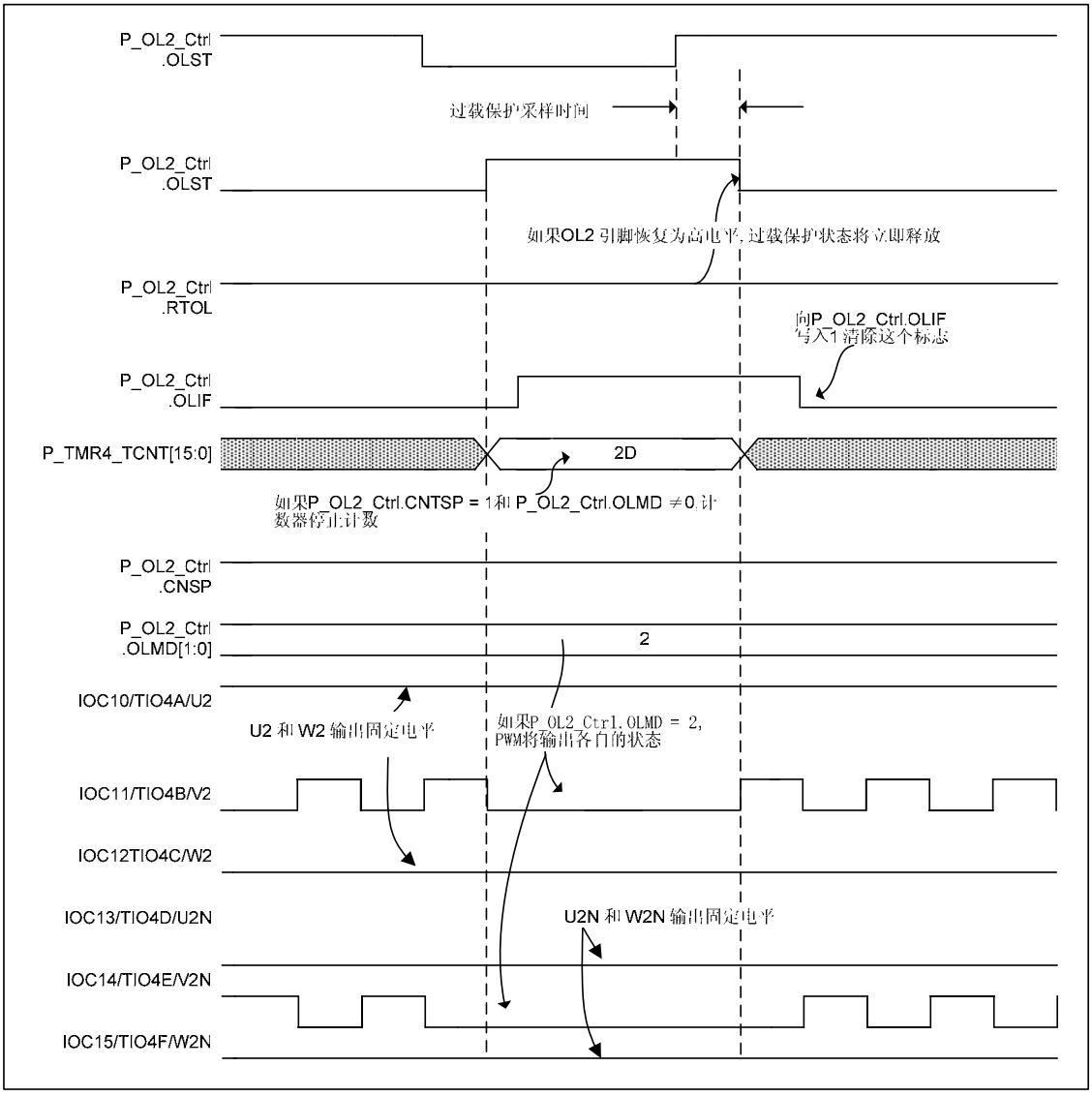


图 13-13 过载 PWM 停止时序

P_OL1_Ctrl($7468)：MCP 过载输入 1 控制和状态寄存器

P_OL2_Ctrl($7469)：MCP 过载输入 2 控制和状态寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	RW	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
OLEN	CNTSP	OLMD		OLST	RTTMB	RTPWM	RTOL

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
OLIE	OLIF	保留		OLCNT			

第 15 位: OLEN：过载保护使能。该位使能/禁止过载保护电路工作。

0 = 禁止

1 = 使能

第 14 位: CNTSP：在过载保护期间停止 PWM 计数器(P_TMR3_TCNT/P_TMR4_TCNT)工作。

0 = 不停止

1 = 停止计数

第 13:12 位 OLMD：在过载保护期间禁止输出相位。

00 =无相位禁止

01 =所有相位都禁止例如：关闭状态

10 =PWM 相位禁止

11 =所有高低相位依照相位有效性被禁止

第 11 位 OLST：过载保护状态。该标志指示了过载保护电路的状态。

0 = 不工作

1 = 在保护状态下

第 10 位 RTTMB：P_TMRx_TGRB (x = 0, 1)寄存器比较匹配后，过载保护释放。

0 = 无动作

1 = P_TMRx_TGRB (x = 0, 1)寄存器比较匹配后，过载保护释放

第 9 位 RTPWM：延迟 1 个周期释放。

0 = 无动作

1 = 当 OL 脚恢复到高电平时，再经过一个周期释放过载保护，当选择此种方式释放过载保护，请将 CNTSP 位设置为不停止计数

第 8 位 RTOL：自动释放过载保护。

0 = 无动作

1 = 当 OL 脚恢复到高电平时，过载保护释放，输出 PWM 信号

第 7 位 **OLIE**：过载中断使能位。

0 = 禁止

1 = 使能

第 6 位 **OLIF**：过载中断标志。该标志表明了过载情况是否已经发生。写“1”清除该标志。

0 = 没发生

1 = 发生过

第 5:4 位 保留

第 3 :0 位 **OLCNT**： 过载保护采样间设置，采样时间为 $F_{CK}/4 * n$（n = 1 到 15）。注意： OLCNT 不能设置为 0。如果将 OLCNT 设置为 0，过载保护电路将认为过载信号一直发生，即使 OL1/2 引脚保持在高状态。如果这时 OLIE 设置为 1，OLIF 标志将不能被清除，过载保护中断程序将一直执行，这将使系统进入不可预知的状态。

13.5.11 MCP 定时器的 PWM 模块写使能控制寄存器

要产生 PWM 波形，用户就必须向 P_TPWM_Write 寄存器写入 0x5A01 来使能 MCP 定时器 3，写入 0x5A02 来使能 MCP 定时器 4。P_TPWM_Write 寄存器用来保证 MCP 定时器 3 和 MCP 定时器 4 不会因 CPU 的异常而被错误的改写和设置。要修改 MCP 定时器 3 和 MCP 定时器 4，必须先将相应的 TMR3/4WE 位置 1。关于在寄存器中 TMR3WE 和 TMR4WE 的功能见下表。建议采用如下的方式操作 P_TPWM_Write 寄存器：首先，读取其值；然后用“0x5A01”或“0x5A02”与读出的值做“或”操作；最后再将操作结果写入 P_TPWM_Write 寄存器中。TMR3WE 和 TMR4WE 各自独立的控制着 TPM 寄存器，具体如下：

TMR3WE：控制着如下寄存器的写操作：

P_TMR3_Ctrl, P_TMR3_IOCtrl, P_TMR3_INT, P_TMR3_Status, P_TMR3_DeadTime, P_TMR_Start, P_TMR_Output

TMR4WE：控制着如下寄存器的写操作：

P_TMR4_Ctrl, P_TMR4_IOCtrl, P_TMR4_INT, P_TMR4_Status, P_TMR4_DeadTime, P_TMR_Start, P_TMR_Output

P_TPWM_Write ($7409)：MCP 定时器/PWM 模块写使能控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0

保留

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R/W	R/W
0	0	0	0	0	0	0	0
保留						TMR4WE	TMR3WE

第 15:2 位　保留

第 1 位　TMR4WE：定时器 4 设置寄存器写使能选择位

0 ＝ 禁止

1 ＝ 使能

第 0 位　TMR3WE：定时器 3 设置寄存器写使能选择位

0 ＝ 禁止

1 ＝ 使能

13.5.12　MCP 定时器完整载入寄存器

在 PWM 输出模式下，为了防止占空比不能同时载入，必须在载入时进行一个更新过程保证正确性。正确的更新过程是：首先更新 P_TMR3/4_TGRA-C，然后将相应的 LDOK 位设置为“1”，一旦 LDOK 位设置为“1”后，就认为所设的占空比参数都已经就绪，并在计数器的值清零时载入到 TGR 中，LDOK 同时清零。在 LDOK 置 1 期间，P_TMR3/4_TGRA-C 中的内容不能通过写操作对其值进行改变。要正确的设置 LDOK 位，需要向 P_TMR_LDOK 的第 7 位到第 2 位之间写入“101010”，否则 LDOK 位无法得到更新。例如，为了将 LDOK0 置 1，必须向 P_TMR_LDOK 写入 0x00A9。

P_TMR_LDOK ($740A)：　MCP 定时器载入完成寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
TLDCHK						LDOK1	LDOK0

第 15:8 位　保留

第 7:2 位　TLDCHK：定时器载入寄存器检查位

要改变 P_TMR_LDOK 的设置，必须向这些位的写入“101010”，否则 LDOK1 和 LDOK0 的值无法改变。这些位读出值为 0。

第 1 位　　LDOK1：P_TMR4_TGRA-C 载入完成位

该位决定着 P_TMR4_TGRA-C 的值是否已经准备好载入 PWM 模块中。该位置 1 后 P_TMR4_TGRA-C 值才可以载入 PWM 模块。载入后，该位会自动清零。注意载该位置 1 期间 P_TMR4_TGRA-C 的值不能由写操作来改变。

第 0 位　　LDOK0：P_TMR3_TGRA-C 载入完成位

该位决定着 P_TMR3_TGRA-C 的值是否已经准备好再入 PWM 模块中。该位置 1 后 P_TMR3_TGRA-C 值才能载入 PWM 模块。载入后，该位会自动清零。注意载该位置 1 期间 P_TMR3_TGRA-C 的值不能由写操作来改变。

13.5.13　定时器启动寄存器

P_TMR_Start 寄存器为 P_TMRx_TCNT (x = 0 - 4)选择定时器的启动/停止操作。当计数器停止工作时，P_TMRx_TCNT (x = 0 - 4)的内容将会被清零。将 TMR3ST 或 TMR4ST 位设置成 1 将会立即启动 P_TMR3_TCNT 或 P_TMR4_TCNT 寄存器工作，反之将将 TMR3ST 或 TMR4ST 位设置成 0 将会立即停止 P_TMR3_TCNT 或 P_TMR4_TCNT 寄存器工作。

P_TMR_Start ($7405)：定时器计数启动寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
保留			TMR4ST	TMR3ST	TMR2ST	TMR1ST	TMR0ST

第 15:5 位　保留

第 4 位　　TMR4ST：MCP 定时器 4 计数启动设置

0 ＝ 停止计数

1 ＝ 开始计数

第 3 位　　TMR3ST：MCP 定时器 3 计数启动设置

0 ＝ 停止计数

1 ＝ 开始计数

第 2 位　　TMR2ST：TPM 定时器 2 计数启动设置

0 ＝ 停止计数

1 ＝ 开始计数

第 1 位　　TMR1ST：PDC 定时器 1 计数启动设置

0 = 停止计数

1 = 开始计数

第 0 位 TMR0ST：PDC 定时器 0 计数启动设置

0 = 停止计数

1 = 开始计数

13.5.14 MCP 定时器的计数寄存器

MCP 定时器 3 和定时器 4 各有一个 TCNT 计数器(P_TMR3_TCNT and P_TMR4_TCNT)，TCNT 计数器是 16 位可读的寄存器，按照输入时钟的进行递增/递减计数，定时控制寄存器中的 TMRPS 位用于选择输入时钟源。P_TMR3_TCNT 和 P_TMR4_TCNT 在中心 PWM 模式下进行递增/递减计数，但在其他模式下只能递增计数。当 TCNT 的值与周期寄存器的值发生匹配时，自动清零。

P_TMR3_TCNT ($7433)：MCP 定时器 3 计数寄存器

P_TMR4_TCNT ($7434)：MCP 定时器 4 计数寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
TMRCNT							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
TMRCNT							

13.5.15 MCP 定时器的通用寄存器

TGRA、TGRB、TGRC、TGRD 是 16 位寄存器。MCP 定时器 3 和 MCP 定时器 4 总共有 8 个通用寄存器，每个定时器 4 个。TGRA、TGRB 和 TGRC 是双功能 16 位可读可写寄存器，用于输出比较或 PWM 寄存器。TGRD 寄存器用于当 TCNT 计数器的值与该寄存器内容匹配时，作为启动 AD 转换的信号。在 TGR 寄存器用于输出比较和 PWM 寄存器时，TGR 和 TCNT 一直在进行相互的比较。 PWM 模式下，无论选择的是边沿 PWM 模式还是中心 PWM 模式，TGR 寄存器都作为占空比寄存器使用，复位后 TGR 寄存器初始化为 0x0000。

P_TMR3_TGRA ($7448)：MCP 定时器 3 通用寄存器 A

P_TMR3_TGRB ($7449)：MCP 定时器 3 通用寄存器 B

P_TMR3_TGRC ($744A)：MCP 定时器 3 通用寄存器 C

P_TMR3_TGRD ($744B)：MCP 定时器 3 通用寄存器 D

P_TMR4_TGRA ($744C)：MCP 定时器 4 通用寄存器 A

P_TMR4_TGRB ($744D)：MCP 定时器 4 通用寄存器 B

P_TMR4_TGRC ($744E)：MCP 定时器 4 通用寄存器 C

P_TMR4_TGRD ($744F)：MCP 定时器 4 通用寄存器 D

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRGLR							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRGLR							

13.5.16 MCP 定时器的缓冲寄存器

缓冲寄存器 TBRx 与通用寄存器 TGRx 构成双缓冲结构，在 PWM 输出时，利用这种结构可以实现多路波形的同步更新。

P_TMR3_TBRA ($7458)：MCP 定时器 3 缓冲寄存器 A

P_TMR3_TBRB ($7459)：MCP 定时器 3 缓冲寄存器 B

P_TMR3_TBRC ($745A)：MCP 定时器 3 缓冲寄存器 C

P_TMR4_TBRA ($745C)：MCP 定时器 4 缓冲寄存器 A

P_TMR4_TBRB ($745D)：MCP 定时器 4 缓冲寄存器 B

P_TMR4_TBRC ($745E)：MCP 定时器 4 缓冲寄存器 C

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRBUF							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRBUF							

13.5.17 MCP 定时器的周期寄存器

P_TMRx_TPR (x = 3, 4)是一个 16 位可读可写寄存器，用于设置定时器或是 PWM 波形的周期。当 P_TMRx_TCNT (x = 3, 4)寄存器的值达到了 P_TMRx_TPR (x = 3, 4)设定值时，P_TMRx_TCNT (x = 3, 4)将会根据 P_TMRx_Ctrl (x = 3, 4)中的模式位“MODE”的设置情况，自动清除(递增计数模式)或开始递减计数(连续递增/递减计数模式)。周期寄存器的默认值是 0xFFFF ，当 P_TMRx_TPR (x = 3, 4) 寄存器设置成 0x0000 时，P_TMRx_TCNT (x = 3, 4)寄存器停止计数并保持在 0x0000。

P_TMR3_TPR ($7438)：MCP 定时器 3 周期寄存器

P_TMR4_TPR ($7439)：MCP 定时器 4 周期寄存器

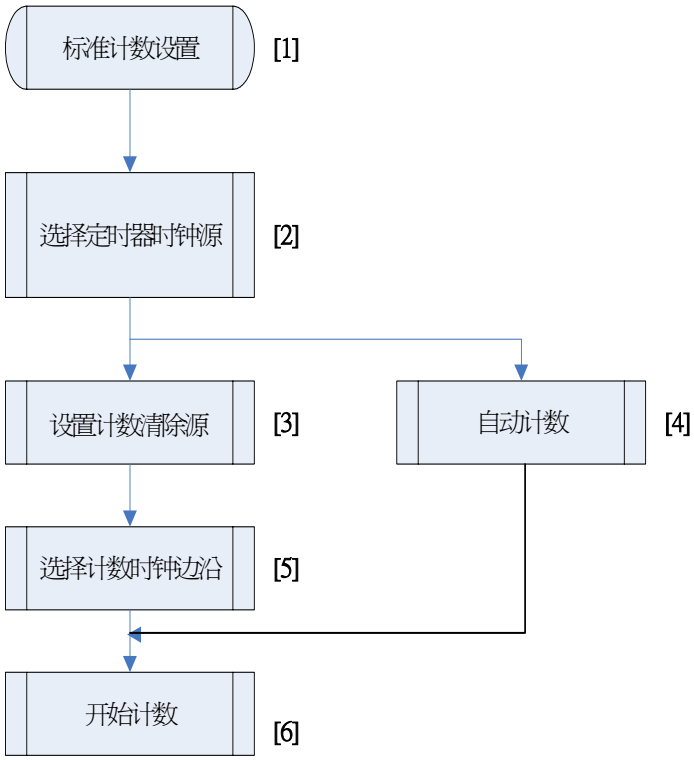
B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
TMRPRD							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
TMRPRD							

13.6 MCP 定时器的操作

13.6.1 标准计数的操作

当 P_TMR_Start 寄存器的 TMR3ST 或 TMR4ST 位置 1 后，相应定时器的 P_TMRx_TCNT (x = 3, 4)寄存器就开始递增计数。计数器以自动计数方式工作，当计数到达 0x0FFFF 后复位为 0x0000，通过将 CCLS 位配置为 111’b 该定时器就变成了一个周期性计数器。计数寄存器在其值达到 P_TMRx_TPR (x = 3, 4)的值后复位为 0x0000，如图 13-14 为标准计数模式的编程流程。



描述：

[1] 将定时器控制寄存器中的 MODE 为设置为 0000’b 。

[2] 通过设置 TMRPS，选择定时器时钟源为内部时钟或外部时钟。

[3] 设置 P_TMRx_TPR (x = 3, 4) 中的 CCLS 为 111’b.

[4] CCLS 采用默认设置 000’b，即禁止进行 TCNT 清除，计数器计数到 0xFFFF 溢出后重新从 0x0000 开始计数。

[5] 在定时器控制寄存器中，通过 CKEGS 和 PRDINT 中断频率来选择计数边沿。

[6] P_TMR_Start 寄存器中的 TMR0ST 或 TMR1ST 置 1，计数器开始工作。如有必要，使能 MCP 定时器周期中断。

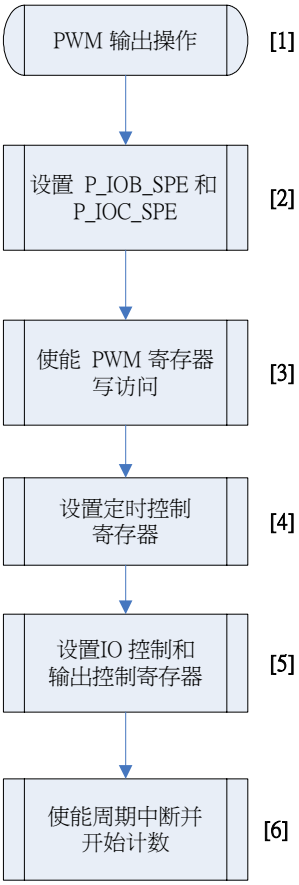
图 13-14 标准计数操作的编程流程举例

P_TMRx_TCNT (x = 3, 4)的初始值是 0x0000 ，P_TMRx_TPR (x = 3, 4 的初始值是 0xFFFF 。当相应位 TMR3ST 或 TMR4ST 置 1 时，P_TMRx_TCNT (x = 3, 4)以自动计数模式计数。

当选择了 P_TMRx_TCNT (x =3, 4)寄存器中的周期比较匹配事件作为寄存器清除源时，定时器以周期性计数操作。通过设置 P_TMRx_TPR (x = 3, 4)寄存器的 CCLS 位为 111’b 来设置周期和计数清除源。设置好后，计数寄存器开始周期性的递增计数操作，此时相应的 TMR3ST 或 TMR4ST 位置 1。当计数器的值与 P_TMRx_TPR (x = 3, 4)相匹配时，定时中断状态寄存器的 TPRIF 标志置 1，同时计数寄存器清零。如果 P_TMRx_INT (x = 3, 4)寄存器中的 TPRIE 位为使能，则发生 MCP 请求中断。

13.6.2 PWM 输出操作

MCP 定时器两个通道来完成 PWM 功能，总计 12 路的输出，输出波形比较匹配时有低有效、高有效，在 TIOxA,、TIOxB、TIOxC、TIOxD、TIOxE 和 TIOxF (x = 3, 4)输出管脚强制高、强制低，用于和 P_TMRx_TGRA、 P_TMRx_TGRB、P_TMRx_TGRC (x = 3, 4)寄存器分别比较匹配，如图 13-15 所示为 PWM 操作的编程流程。



描述：

[[1] 如有必要，在 PWM 特殊功能下的 IO 端口设为输出和错误输入。

[2] 使能PWM 的特殊功能寄存器P_IOB_SPE 和 P_IOC_SPE.

[3] 向 P_TPWM_Write 寄存器写入控制字使能写访问。

[4] 设置 CCLS 位为 111’b，使得周期寄存器确定了计数周期和计数清除源。

[5] 通过 P_TMRx_IOCtrl (x = 3, 4) 寄存器选择比较匹配输出模式，通过 P_TMRx_OutputCtrl (x = 3, 4)寄存器设定 PWM 波形。

[6] P_TMR_Start 寄存器的 TMR3ST 或 TMR4ST 置 1，使能周期中断，计数器开始工作。

图 13-15　PWM 操作的编程流程举例

13.7 设计参考

【例 13-1】怎样设置定时器 3 的频率？

定时器 3 的频率与定时器 3 的时钟源，计数时钟边沿以及由 TMRPS、CKEGS 和 PRDINT 位所设定的周期中断频率有关。定时器 3 有 8 个中断源，分别是 Fck/1、Fck/4、Fck/16、 Fck/64、Fck/256、Fck/1024、TCLKA 和 TCLKB。CKEGS 位对输入时钟边沿进行选择。当输入时钟以双沿计数时，输入时钟周期将会减半。当选定 FCK/1 位计数器时钟时，如果选择了双沿计数模式，计数器将以上升沿计数。PRDINT 位的设置将会影响定时器 3 周期中断的发生频率，将这些设置做好后，要写入 P_TMR3_PRD 数据。

例如，如果要将定时器 3 周期中断频率设置成 8K Hz：**P_TMR3_TPR = (source clock / 8000) – 1**

```
/* MCP 定时器 3 中断频率为 8000 Hz ，  */
/*  选择 MCP3  时钟源为 Fck/1 = 20MHz. */
#include "mc75Regs.h"
int main(void)
{
Disable_FIQ_IRQ();
P_TPWM_Write->W = 0x5A03;                              /* MCP3 修改使能*/
```

```
P_TMR3_Ctrl->B.CCLS = CB_TMR3_CCLS_PTR;          /* TPR 比较匹配时清除 */
P_TMR3_Ctrl->B.TMRPS = CB_TMR3_TMRPS_FCKdiv1;
                                                 /* 计数频率为 FCK/1,上升沿 */
P_TMR3_Ctrl->B.CKEGS = CB_TMR0_CKEGS_Rising;  /* 上升沿计数 */
P_TMR3_Ctrl->B.MODE = CB_TMR3_MODE_Normal;    /* 标准计数模式*/
P_TMR3_Ctrl->B.PRDINT = CB_TMR3_PRDINT_Period;/* 每周期中断 */

P_TMR3_TPR->W = 2499;                            /*设置计数周期 2499，频率 8000 Hz
                                                 */
P_TMR3_INT->B.TPRIE = 1;                         /* 使能 MCP3 中断 */
P_TMR_Start->W = CW_TMR_TMR3ST_Start;            /* 启动 MCP3 */
INT_FIQ_IRQ();                                   /* 使能 FIQ/IRQ 中断 */
while(1);
return (1);
}
/*****************************************************************************/
void IRQ3(void) __attribute__ ((ISR));
void IRQ3(void)
{
if(P_TMR3_Status->B.MCP3IF)
{
P_TMR3_Status->W = CW_TMR3_TPRIF_Enable;         /* 清除 MCP3 的 TPRIF 标志*/
/* 执行 TMR3_TPR ISR 任务 */
}
}
```

【例 13-2】怎样产生驱动 120 度 6 步波形驱动直流无刷电机?

首需要设置 MCP 定时器 3 的 TPR 中断频率位。120 度 6 步波形是由电机内的 PDC 定时器模块的霍尔位置信号的反馈信号决定的，见第 11 章。定时器 3 的中断频率是 8000 Hz，选择 $F_{CK}/1$ = 24MHz 作为定时器 3 时钟源。三相占空比寄存器只用到了 TGRA 寄存器。没有插入死区控制。

```
#include "Spmc75_regs.h"
#include "unspmacro.h"

int main(void)
{
Disable_FIQ_IRQ();
Init_TMR3 ();                                          /* 初始化 MCP3 模块 */

P_IOB_SPE->W = 0x003F;                                 /* 使能 UVW-1 输出特
                                                       殊功能
*/ P_TMR3_IOCtrl->W = 0x0111;                          /* TIO3A - TIO3F 输出
                                                       使能*/
P_TMR3_OutputCtrl->B.DUTYMODE = 0;
/*三相共用 TGRA 寄存器 */

P_TMR3_OutputCtrl->B.POLP = CB_TMR3_POLP_Active_High;  /* 高有效*/
P_POS0_DectCtrl->W = 0x0000;                           /* 采样时钟 = FCK/4 */
P_POS0_DectCtrl->B. SPLMOD = 1;                        /* 周期采样*/
P_POS0_DectCtrl->B.PDEN = 1;                           /* 使能侦测位置功能 */

P_TMR_Start->W = CW_TMR_TMR3ST_Start;                  /* 启动 MCP3 */
INT_FIQ_IRQ();                                         /* 使能 FIQ/IRQ 中断 */

while(1);
return (1);
}
/*****************************************************************************/
void IRQ3(void) __attribute__ ((ISR));
void IRQ3(void)
{
int position;
if(P_TMR3_Status->B.TPRIF)
{
P_TMR3_Status->W = CW_TMR3_TPRIF_Enable;               /*清除 MCP3 TPRIF */
 /* 依据侦测位置信号，产生 120 度 6 路 PWM 波形，如下的条件分支可根据开发者应用而有
所不同*/
position = P_POS0_DectData->W;
switch(position)
```

```
{
case (1):
P_TMR3_OutputCtrl->B.UPWM = 0;
P_TMR3_OutputCtrl->B.UOC = 0;                    /* U1 = L, U1N = L */
 P_TMR3_OutputCtrl->B.VPWM = 0;
P_TMR3_OutputCtrl->B.VOC = 1;                    /* V1 = L, V1N = H */

P_TMR3_OutputCtrl->B.WPWM = 1;
P_TMR3_OutputCtrl->B.WOC = 2;                    /* W1 = PWM, W1N = L */
break;

case (2):
P_TMR3_OutputCtrl->B.UPWM = 1;
P_TMR3_OutputCtrl->B.UOC = 2;                    /* U1 = PWM, U1N = L */

P_TMR3_OutputCtrl->B.VPWM = 0;
P_TMR3_OutputCtrl->B.VOC = 1;                    /* V1 = L, V1N = H */

P_TMR3_OutputCtrl->B.WPWM = 0;
P_TMR3_OutputCtrl->B.WOC = 0                     /* W1 = L W1N = L */
break;

case (3):
P_TMR3_OutputCtrl->B.UPWM = 1;
P_TMR3_OutputCtrl->B.UOC = 2;                    /* U1 = PWM, U1N = L */

P_TMR3_OutputCtrl->B.VPWM = 0;
P_TMR3_OutputCtrl->B.VOC = 0;                    /* V1 = L, V1N = L */

P_TMR3_OutputCtrl->B.WPWM = 0;
P_TMR3_OutputCtrl->B.WOC = 1;                    /* W1 = L, W1N = H */
break;

case (4):
P_TMR3_OutputCtrl->B.UPWM = 0;
P_TMR3_OutputCtrl->B.UOC = 0;                    /* U1 = L, U1N = L */

P_TMR3_OutputCtrl->B.VPWM = 1;
P_TMR3_OutputCtrl->B.VOC = 2;                    /* V1 = PWM, V1N = L */

P_TMR3_OutputCtrl->B.WPWM = 0;
P_TMR3_OutputCtrl->B.WOC = 1;                    /* W1 = L, W1N = H */
break;
```

```
case (5):
P_TMR3_OutputCtrl->B.UPWM = 0;
P_TMR3_OutputCtrl->B.UOC = 1;                    /* U1 = L, U1N = H */

P_TMR3_OutputCtrl->B.VPWM = 1;
P_TMR3_OutputCtrl->B.VOC = 2;                    /* V1 = PWM, V1N = L */

P_TMR3_OutputCtrl->B.WPWM = 0;
P_TMR3_OutputCtrl->B.WOC = 0;                    /* W1 = L, W1N = L */
break;

case (6):
P_TMR3_OutputCtrl->B.UPWM = 0;
P_TMR3_OutputCtrl->B.UOC = 1;                    /* U1 = L, U1N = H */

P_TMR3_OutputCtrl->B.VPWM = 0;
P_TMR3_OutputCtrl->B.VOC = 0;                    /* V1 = L, V1N = L */

P_TMR3_OutputCtrl->B.WPWM = 1;
P_TMR3_OutputCtrl->B.WOC = 2;                    /* W1 = PWM, W1N = L */
break;
}

P_TMR3_TGRA->W = 750;                            /* 设置占空比 = 30 % 三相 */
P_TMR_LDOK->W = CW_TMR_LDOK0;                    /* TGRA - TGRC 同时载入 */
}
}
```

【例 13-3】在 U1/U1N, V1/V1N 和 W1/W1N 功率管两极如何插入死区控制?

死区控制的作用是保护三相变频因短路造成的损害。要加入死区控制，需要对 P_TMRx_DeadTime (x = 3, 4)寄存器进行恰当的配置，还要将 P_TMRx_OutputCtrl (x = 3, 4)的 UOC/VOC/WOC 位选择成 PWM 补偿模式。以下是三相变频电源应用的编程实例：

```
#include "Spmc75_regs.h"
#include "unspmacro.h"

int main(void)
{
Disable_FIQ_IRQ();
Init_TMR3();                                     /* 初始化 MCP3 模块 */

P_IOB_SPE->W = 0x003F;                           /* 使能 UVW-1 输出特殊功能 */
P_TMR3_IOCtrl->W = 0x0111;                       /* TIO3A - TIO3F 输出使能 */
```

```
    P_TMR3_OutputCtrl->B.DUTYMODE = 0;
                                                /* 三相共用 TGRA 寄存器 */
    P_TMR3_OutputCtrl->B.POLP = CB_TMR3_POLP_Active_High;    /* 高有效 */
    P_TMR_Start->W = CW_TMR_TMR3ST_Start;                    启动 MCP3 */
    INT_FIQ_IRQ();                                          FIQ/IRQ 中断 */

    while(1);
    return (1);
    }
    /*****************************************************************************/
    void IRQ3(void) __attribute__ ((ISR));
    void IRQ3(void)
    {
    if(P_INT_Status->B.MCP3IF)
    {
    P_TMR3_Status->W = CW_TMR3_TPRIF_Enable;        /* 清除 MCP3 的 TPRIF 标志 */
    P_TMR3_DeadTime->B.DTWE = 1;                    /* 使能 W 相的死区定时器 */
    P_TMR3_DeadTime->B.DTVE = 1;                    /* 使能 V 相的死区定时器 */
    P_TMR3_DeadTime->B.DTUE = 1;                    /*使能 U 相的死区定时器*/
    P_TMR3_DeadTime->B.DTP = 12;
                                                    /*时钟频率 Fck/4, FCK = 20.0 MHz
                                                    时,设置死区时间为 2.0us*/
                                                    /* PWM 在 U 相 PWM 输出中插入
                                                    死区控制的范例 */
    P_TMR3_OutputCtrl->W = (P_TMR3_OutputCtrl->W & 0xF800) | \
    (CW_TMR3_POLP_1_UP_L_UN_L | CW_TMR3_POLP_1_VP_L_VN_H | \
CW_TMR3_POLP_1_WP_PWM_WN_CPWM);
    }
    }
```

【例 13-4】 设置错误输入 1 功能保护的 MCP3 6 路 PWM 输出。以下是编程范例：

```
    #include "Spmc75_regs.h"
    #include "unspmacro.h"

    int main(void)
    {
    Disable_FIQ_IRQ();
    Init_TMR3();                                    /* 初始化 MCP3 模块 */

    P_IOB_SPE->W = 0x003F;                          /* 使能 UVW-1 输出特殊功能 */
    P_TMR3_IOCtrl->W = 0x0111;                      /* TIO3A - TIO3F 输出使能 */
    P_TMR3_OutputCtrl->B.DUTYMODE = 0;
                                                    /* 三相共用 TGRA 寄存器 */
    P_TMR3_OutputCtrl->B.POLP = CB_TMR3_POLP_Active_High;    /* 高有效 */
```

```
P_IOB_SPE->W |= CW_IOB_FTIN1_SFR_EN;      /* 使能 FTIN1 特殊功能 */
P_Fault1_Ctrl->B.OCE = 1;                 /* 使能输出电平比较 */
P_Fault1_Ctrl->B.OCIE = 1;                /* 使能输出比较中断 */

P_Fault1_Ctrl->B.OCLS = 1;                /* 高电平比较 */
P_Fault1_Ctrl ->B.FTPINE   = 1;           /* 使能 FTIN1 引脚 */
P_Fault1_Ctrl->B.FTPINIE = 1;             /* Fault 1 中断使能 */
P_Fault1_Ctrl->B.FTCNT = 1;
P_INT_Priority->B.FTIP = 1;               /* FTIN1 分配给 FIQ */

while(1);
return (1);
}
/*****************************************************************************/
void FIQ(void) __attribute__ ((ISR));
void FIQ(void)
{
if(P_INT_Status->B.FTIF)
{
if(P_Fault1_Ctrl->B.OSF)
{
P_Fault1_Release->W = 0xAA55;                  /* 清除 OSF 标志 */
P_Fault1_Release->W = 0x55AA;
P_Fault1_Ctrl->B.OCE = 1;                      /* 再次使能 OCE 引脚 */
}

if(P_Fault1_Ctrl->B.FTPINIF)
{
P_Fault1_Release->W = 0x55AA;                  /* 清除 FTPINIF 标志 */
P_Fault1_Release->W = 0xAA55;
P_Fault1_Ctrl->B.FTPINE   = 1;                 /* 再次使能 FTIN1 引脚 */
}
}
}
```

【例 13-5】启动 MCP3 过载保护功能。

```
void init_OLP3(void)
{
P_OL1_Ctrl->B.OLEN = 1;                  /* 使能过载保护 */
P_OL1_Ctrl->B.OLMD = 2;                  /* 当发生过载时禁止 PWM 相位 */
P_OL1_Ctrl->B.OLCNT = 1;                 /* 过载采样定时计数器 = 1, 时钟 Fck/4 */
    P_OL1_Ctrl->B.RTOL = 1;   //
}
```

```
/*****************************************************************************/
void IRQ3(void) __attribute__ ((ISR));
void IRQ3(void)
{
if(P_INT_Status->B.MCP3IF)
{
P_TMR3_Status->W = CW_TMR3_TPRIF_Enable;  /* 清除 MCP3 TPRIF 标志 */

if(P_OL1_Ctrl->B.OLST)                              /* 过载保护发生? */
{
//过载事件处理
}
}
}
}
```

【例 13-6】设置 PWM 同步使能位释放过载保护

```
void init_OLP3(void)
{
P_OL1_Ctrl->W = (CW_TMR3_OPR_OLEN | CW_TMR3_OPR_OLMD_PWMDis | \
CW_TMR3_OPR_RTPWM_Enable | CW_TMR3_OPR_OLEN);
P_OL1_Ctrl->B.OLCNT = 1;
                                        /* 过载保护使能*/
                                        /*发生过载时禁止所有相位输出*/
                                        /* PWM 使能，退出过载保护 */
                                        /* 过载采样定时计数器 = 1 ，时钟 FCK/4 */
}
```

【例 13-7】在正弦交流感应电机中 MCP3 寄存器的建立的过程

SPMC75X 系列微控制器支持沿和中心 PWM 波形发生。在本例中，使用中心 PWM 模式，因其较少谐波的比较适合于交流感应电机的驱动。

PWM 的三相脉冲宽度是由 P_TMR3_TGRA、P_TMR3_TGRB 和 P_TMR3_TGRC 决定的，正弦波形数据预先写入存储器中，通过查表向这些寄存器写入相应，产生相应的正弦波形。如果定时器通用寄存器 TGR 与 P_TMR3_TCNT 定时计数器匹配，相应的端口就会根据 P_TMR3_OutputCtrl 寄存器触发输出事件。在高低相位之间的死区控制是通过 P_TMR3_DeadTime 决定的，见图 13-6 的参考波形图。

相比较三相交流变频正弦波，锯齿波的频率是 PWM 的载波频率，这是由 CPU 时钟频率 F_{CK} 和周期寄存器 P_TMR3_TPR 决定的。一系列不同脉冲宽度的 PWM 波形可以组合成一个正弦波。以下的流程图是交流感应电机的正弦驱动设置过程。

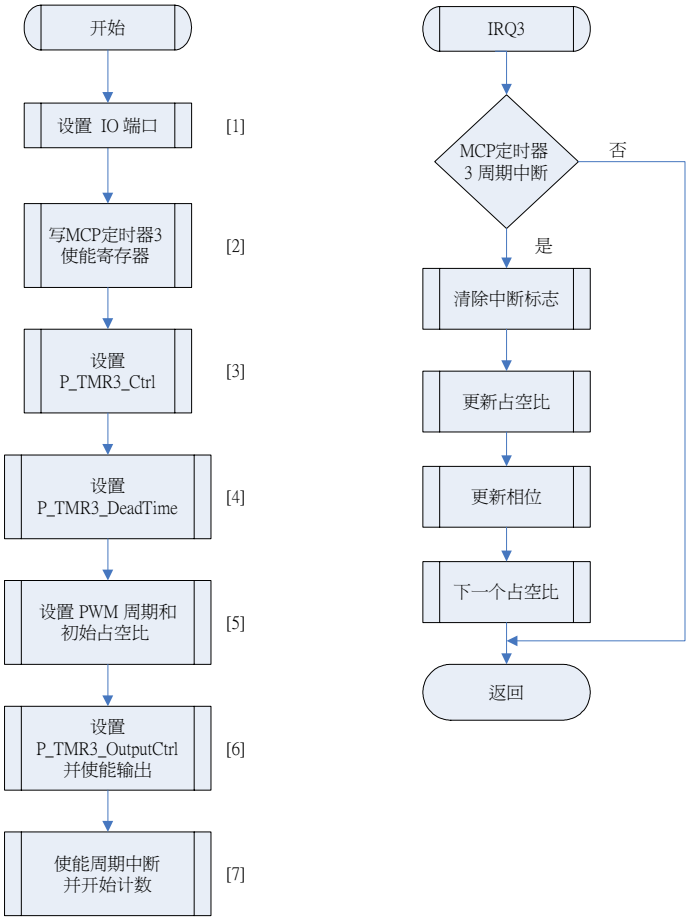


图 13-16 正弦 PWM 操作的编程流程图举例

描述：

[1] 为 PWM 输出端口设置 P_IOB_Dir 的值并使能 P_IOB_SPE 的特殊功能

[2] 写 MCP 定时器 3 使能寄存器：P_TPWM_Write->W= P_TPWM_Write->W | 0x5A01;

[3] 设置 MCP 定时器 3 控制寄存器 P_TMR3_Ctrl：

PRDINT = 0, 每周期产生中断并且定时器的周期由 P_TMR3_TPR 和 CPU 运行时钟频率决定。

MODE = 1010’b, 设置为中心 PWM 模式。

CCLS = 111’b, 设置 MCP 定时器 3 的清除源。.

CKEGS = 0, 上升沿计数

TMRPS = 0, 时钟源为 FCK

[4] 将 P_TMR3_DeadTime 设置为 0x7001，使能三相死区控制。

[5] 置 PWM 周期和初始占空比值

P_TMR3_TPR->W = 0x0960,决定 PWM 周期

P_TMR3_TGRA->W = 0x0400, U 相占空比值

P_TMR3_TGRB->W = 0x0766, V 相占空比值

P_TMR3_TGRC->W = 0x0088, W 相占空比值

P_TMR_LOAD->W = 0x00A9, 同步载入

[6] 设置 MCP 定时器 3 的 PWM 输出模式并使能相应输出引脚。

P_TMR3_OutputCtrl->W = 0x8700, 三相独立 PWM 输出。

P_TMR3_IOCtrl->W = 0x0111, 初始化为 0，比较匹配时输出 1

P_TMR_Output->W = 0x003F, 使能 MCP 定时器 3 PWM 输出

[7] 使能 MCP 定时器 3 周期中断和启动计数。

P_TMR3_INT->B.TPRIE = 1, 使能周期中断。

P_TMR_Start->B.TMR3ST = 1, 定时器 3 开始计数。

以下代码为以上描述的所有操作的编程过程。

```
#include "Spmc75_regs.h"
#include "Spmc_typedef.h"
#include "unspMACRO.h"
extern int Sin_TAB[];
UInt16 SPWM_phases;                          /* 相位增量 */
UInt16 Phases_Temp;                          /* 查表索引 */
UInt16 Phases_Add_Data;                      /* 相位位移 */
int NEW_Data[3];
void Sys_Init(void);
void CCP_Init(void);

int main(void)
{
    Sys_Init();
    CCP_Init();
    INT_IRQ();
    while(1);
}

/*****************************************************************************/
void Sys_Init(void)
{
      while(P_Clk_Ctrl->B.OSCSF & 1)         /* 查询振荡器状态标志 */
      {
          P_Clk_Ctrl->B.OSCSF = 1;           /* 清除振荡器状态标志 */
      }
     P_System_Option->W = 0x5551             /* 使用外部晶振 */
SPWM_phases = 0;
Phases_Add_Data = 656;                       /* 参考频率 = 50Hz */
}
/*****************************************************************************/
void CCP_Init(void)
```

```
{
    P_IOB_Dir->W = 0x0000;                  /* IO 设置 */
P_IOB_Attrib->W = 0x0FFFF;
    P_IOB_Data->W = 0xff00;
    P_IOB_SPE->W = 0x00ff;
    P_TPWM_Write->W = 0x5a01;               /* 使能控制寄存器写功能 */
    P_TMR3_OutputCtrl->W = 0x8700;          /* 3 相独立输出控制 */
    P_TMR3_IOCtrl->W = 0x0111;
    P_TMR3_DeadTime->W = 0x7001;            /* 选择死区时间 */
    P_TMR3_Ctrl->B.PRDINT = 0;
    P_TMR3_Ctrl->B.MODE = 10;
    P_TMR3_Ctrl->B.CCLS = 7;
    P_TMR3_Ctrl->B.CKEGS = 0;
    P_TMR3_Ctrl->B.TMRPS = 0;
    P_TMR_Output->W = 0x3f3f;               /* 使能输出引脚 */
P_TMR3_TPR->W = 0x0960;                     /* 定时器周期设置 */
P_TMR3_TGRA->W = 0x0400;
 P_TMR3_TGRB->W = 0x0766;
P_TMR3_TGRC->W = 0x0088;
P_TMR3_INT->W = 0x0010;                     /* 定时器周期中断使能*/
    P_TMR_Start->B.TMR3ST = 1;              /* MCP3 运行*/
}

/*****************************************************************************/
void IRQ3(void) __attribute__ ((ISR));
void IRQ3(void)
{
    if(P_TMR3_Status->B.TPRIF)
    {
        P_TMR3_Status->B.TPRIF = 1;         /* 清除 TPRIF 标志 */
        P_TMR3_TGRA->W = NEW_Data[0];       /* 更新数据 */
        P_TMR3_TGRB->W = NEW_Data[1];
        P_TMR3_TGRC->W = NEW_Data[2];
        P_TMR_LDOK->W = CW_TMR_LDOK0;
        SPWM_phases += Phases_Add_Data;     /* 相位递增 */
        Phases_Temp = SPWM_phases>>6
        NEW_Data[0] = Sin_TAB[Phases_Temp];
        NEW_Data[1] = Sin_TAB[(Phases_Temp+341)&0x03ff];
        NEW_Data[2] = Sin_TAB[(Phases_Temp+683)&0x03ff];
    }
}
```

第14章 比较匹配定时器

14.1简介

SPMC75X 系列微控制器内嵌两个 16 位的比较定时器(CMT)。当计数到设定值后产生中断。图 14-1 显示比较定时器(CMT)时序。

- 可选择 8 个计数器输入时钟
  - $F_{CK}/1$, $F_{CK}/2$, $F_{CK}/4$, $F_{CK}/8$, $F_{CK}/16$, $F_{CK}/64$, $F_{CK}/256$, $F_{CK}/1024$
- 中断源
  - 每个通道都具有其独立的比较匹配中断

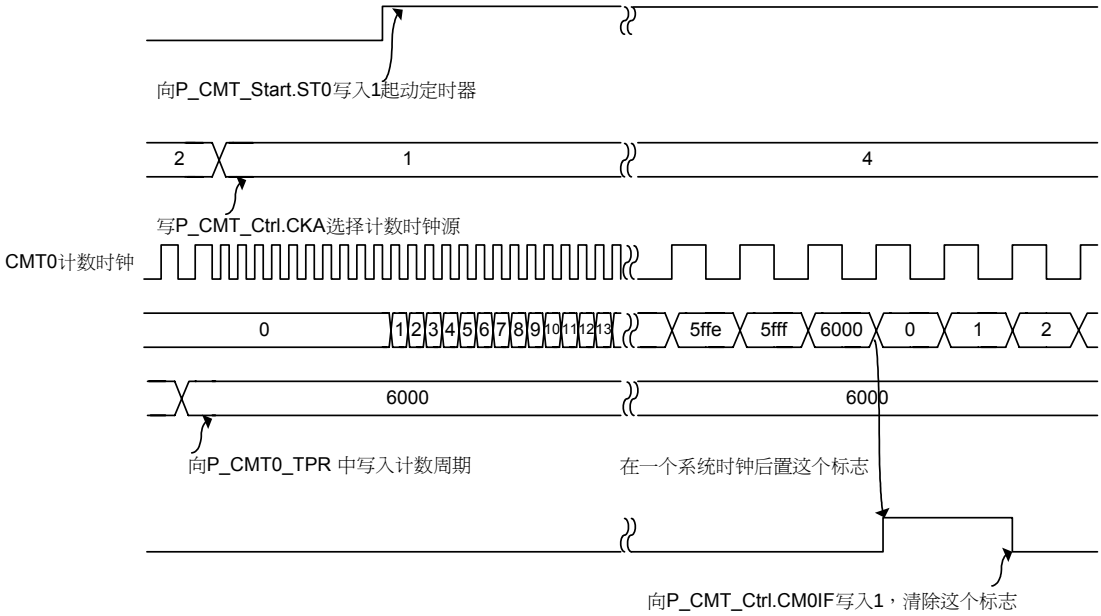


图 14-1 比较定时器(CMT)时序

14.2控制寄存器

SPMC75X 系列微控制器内嵌的比较定时器(CMT)模块共有 6 个控制寄存器，如表 14-1 所示。通过这 6 个控制寄存器可以完成 CMT 模块所有功能的控制。

表 14-1 比较匹配定时器寄存器

地址	寄存器	名称
7500h	P_CMT_Start	比较匹配定时器启动控制寄存器
7501h	P_CMT_Ctrl	比较匹配定时控制与状态寄存器
7508h	P_CMT0_TCNT	比较匹配定时器 0 计数寄存器
7509h	P_CMT1_TCNT	比较匹配定时器 1 计数寄存器
7510h	P_CMT0_TPR	比较匹配定时器 0 周期寄存器
7511h	P_CMT1_TPR	比较匹配定时器 1 周期寄存器

14.2.1 比较匹配定时器起动寄存器

P_CMT_Start ($7500)：比较匹配定时器起动寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R/W	R/W
0	0	0	0	0	0	0	0
保留						ST1	ST0

第 15:2 位： 保留

第 1 位： ST1：比较匹配定时器 1 计数开始位

0 ＝ P_CMT1_TCNT 停止计数并清零为 0x0000

1 ＝ P_CMT1_TCNT 启动计数

第 0 位： ST0：比较匹配定时器 1 计数开始位

0 ＝ P_CMT0_TCNT 计数器运行停止并清零为 0x0000

1 ＝ P_CMT0_TCNT 计数器运行使能

14.2.2 比较匹配定时器控制和状态寄存器

P_CMT_Ctrl ($7501)：比较匹配定时器控制和状态寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R	R	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CM1IF	CM1IE	保留			CKB		

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R	R	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CM0IF	CM0IE	保留			CKA		

第 15 位 CM1IF：CMT1 比较匹配中断标志。此标志指示出 P_CMT1_TCNT 与 P_CMT1_TPR 的值是否匹配。写入‘1’可清除此标志。

0 ＝ 不匹配

1 = 匹配

第 14 位 **CM1IE**：CMT1 比较匹配中断使能

0 = CMT1 比较匹配中断禁止

1 = CMT1 比较匹配中断使能

第 13:11 位 保留

第 10:8 **CKB**：CMT1 时钟选择位

000 = F_{CK} /1

001 = F_{CK}/2

010 = F_{CK}/4

011 = F_{CK}/8

100 = F_{CK}/16

101 = F_{CK}/64

110 = F_{CK}/256

111 = F_{CK} /1024

第 7 位 **CM0IF**：CMT0 比较匹配中断标志。此标志指示出 P_CMT0_TCNT 与 P_CMT0_TPR 的值是否匹配。写入‘1’可清除此标志。

0 = 不匹配

1 = 匹配

第 6 位 **CM0IE**：CMT0 比较匹配中断使能

0 = CMT0 比较匹配中断禁止

1 = CMT0 比较匹配中断使能

第 5:3 位 保留

第 10:8 位 **CKA**：CMT0 时钟选择位

000 = F_{CK}/1

001 = F_{CK}/2

010 = F_{CK}/4

011 = F_{CK}/8

100 = F_{CK}/16

101 = F_{CK}/64

110 = F_{CK}/256

111 = F_{CK}/1024

14.2.3 比较匹配定时器的计数器

P_CMT0_TCNT ($7508)：比较匹配定时器 0 计数寄存器

P_CMT1_TCNT ($7509)：比较匹配定时器 1 计数寄存器

比较匹配定时计数器是一个 16 位递增计数器，初始值为 0x0000。在定时器停止或向周期寄存器 P_CMTx_TPR (x = 0, 1)中写入新值时自动清零。

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
CMTCNT							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
CMTCNT							

14.2.4 比较匹配定时器的周期寄存器

P_CMT0_TPR ($7510)：比较匹配定时器 0 周期寄存器

P_CMT1_TPR ($7511)：比较匹配定时器 1 周期寄存器

比较匹配定时器周期寄存器 P_CMTx_TPR (x = 0, 1)是一个用于设定比较匹配周期的 16 位寄存器，初始值为 0x0000。当向 P_CMTx_TPR (x = 0, 1)中写入一个新值时，P_CMTx_TCNT (x = 0, 1)就自动清零。

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CMTPR							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CMTPR							

14.3设计参考

【例 14-1】将 CMT0 和 CMT1 两个比较匹配定时器通道的频率各自设定为 1000Hz 和 5Hz。

```
    void Init_CMT (void)
    {
    P_CMT0_TPR->W = P_CMT1_TPR->W = 0x0000;
    P_CMT_Start->W = 0x0000;                          /* 停止所有比较匹配定时器 */
    P_CMT_Ctrl->W = 0x0000;                           /* 复位设置 */

    P_CMT_Ctrl->B.CKA = 0;                          /* CMT0 时钟选择 = FCK / 1 */
P_CMT_Ctrl->B.CM0IE = 1;                            /* CMT0 比较匹配中断使能 */
    P_CMT_Ctrl->B.CKB = 7;                          /* CMT1 时钟选择 = FCK / 1024 */
P_CMT_Ctrl->B.CM1IE = 1;                            /* CMT1 比较匹配中断使能*/
    P_CMT0_TPR->W =   19999;                        /* CMT0 频率 1000 Hz ，时钟
                                                    FCK = 20.0 MHz */
    P_CMT1_TPR->W =   3905;                         /* CMT1 频率 5 Hz ，时钟 FCK =
                                                    20.0 MHz */

    P_CMT_Start->B.ST0 = 1;                         /* 启动 CMT0 计数器 */
    P_CMT_Start->B.ST1 = 1;                         /*启动 CMT1 计数器*/
    }
    /*****************************************************************************/
    void IRQ7(void) __attribute__ ((ISR));
    void IRQ7(void)
    {
    if(P_INT_Status->B.CMTIF)
    {
    if(P_CMT_Ctrl->B.CM0IF)
    {
    P_CMT_Ctrl->B.CM0IF = 1;                        /* 清除 CM0IF 标志 */
    // place your codes here.
    }

    if(P_CMT_Ctrl->B.CM1IF)
    {
    P_CMT_Ctrl->B.CM1IF = 1;                        /* 清除 CM0IF 标志*/
    // place your codes here.
    }
    }
    }
```

例 14-1 比较匹配定时器设计参考

第15章 时基模块和蜂鸣器模块

15.1简介

SPMC75X 系列微控制器内嵌一个时基模块，此模块包含一个 16 位的计数器，可以产生 $F_{CK}/2$～$F_{CK}/65536$ 参考时钟供芯片外设使用。可向时基复位寄存器(P_TMB_Reset)写入 0x5555 清空时基计数器。通过时基模块，可以产生 50%占空比的脉冲，以驱动蜂鸣器。被选通的时基信号接入 IOC4/BZO 引脚。时基和蜂鸣器输出时序如图 15-1 所示

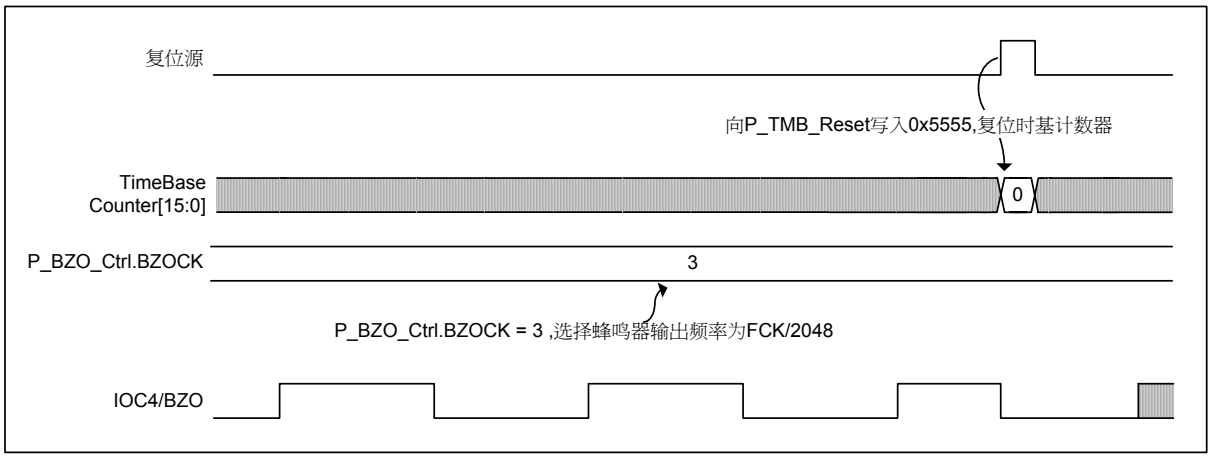


图 15-1 时基和蜂鸣器输出时序

15.2控制寄存器

SPMC75X 系列微控制器内嵌的时基模块和蜂鸣器模块共有 2 个控制寄存器，如表 15-1 所示。通过这 2 个控制寄存器可以完成时基模块和蜂鸣器模块所有功能的控制。

表 15-1 时基模块和蜂鸣器模块寄存器

地址	寄存器	名称
70B8h	P_TMB_Reset	时基复位寄存器
70B9h	P_BZO_Ctrl	蜂鸣器输出控制寄存器

15.2.1 时基复位寄存器

向时基复位寄存器写入$5555H。时基计数器将复位，时基计数器清零。

P_TMB_Reset ($70B8)：时基复位寄存器

B15	B14	B13	B12	B11	B10	B9	B8
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
TBRR							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
TBRR							

15.2.2 蜂鸣器输出控制寄存器

这个寄存器用来使能蜂鸣器输出功能和选择蜂鸣器驱动频率。

P_BZO_Ctrl ($70B9)：蜂鸣器输出控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
BZOEN	保留						

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R/W	R/W
0	0	0	0	0	0	0	0
保留						BZOCK	

第 15 位　　BZOEN：蜂鸣器输出使能位

0＝禁止

1＝使能

第 14:2 位　保留

第 1:0 位　　BZOCK：蜂鸣器输出频率设置位，如表 15-2 所示

表 15-2　蜂鸣器输出频率 BZOCK 表

BZOCK	蜂鸣器输出频率	当 F_{CK}=24MHz 时
00	F_{CK} /16384	1.464kHz
01	F_{CK} /8192	2.929kHz
10	F_{CK} /4096	5.859kHz
11	F_{CK} /2048	11.718kHz

15.3 设计参考

【例 15-1】设置输出时钟频率为 1500 Hz @ FCK = 24.0 MHz 的蜂鸣器单元。

```
P_BZO_Ctrl->W = 0x0000;                          /* 复位蜂鸣器并禁用 */
P_BZO_Ctrl->B.BZOCK = 0;
/* 蜂鸣器时钟频率 = FCK / 16384 */
P_BZO_Ctrl->B.BZOEN = 1;                         /* 使能蜂鸣器时钟输出 */
```

例 15-1 蜂鸣器输出控制设定

第16章 SPI 标准外设接口

16.1简介

SPMC75X 系列微控制器内嵌一个 SPI 通信模块，支持主从两种工作模式（通过对 P_SPI_Ctrl 寄存器的 SPIMS 位的设定来选择），最高可以到 6M 的通信速度。在从模式下，建议将从机系统时钟频率至少设置为 SPICLK 信号的频率两倍，以保证该模式运行正常。

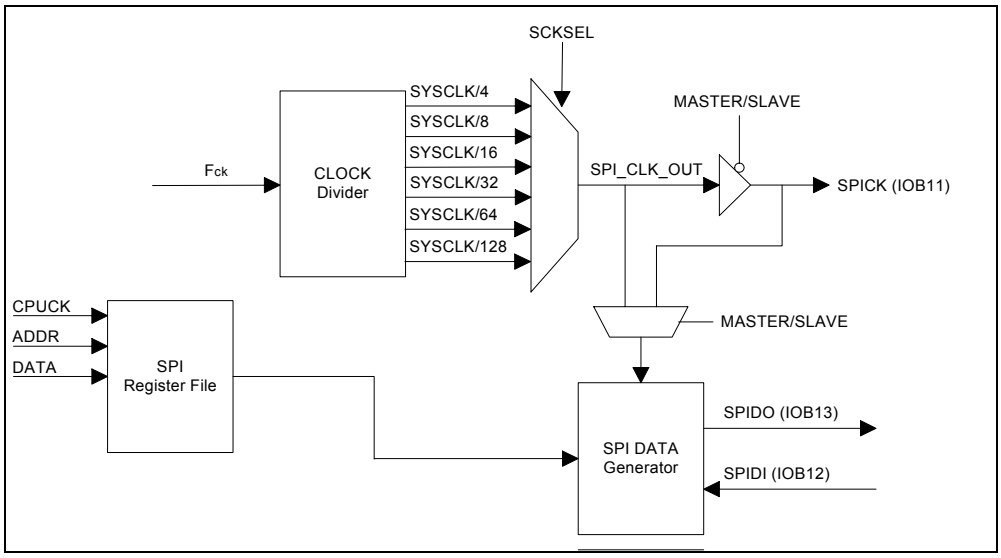


图 16-1 SPI 结构框图

图 16-2 是 SPI 模块的功能框图。

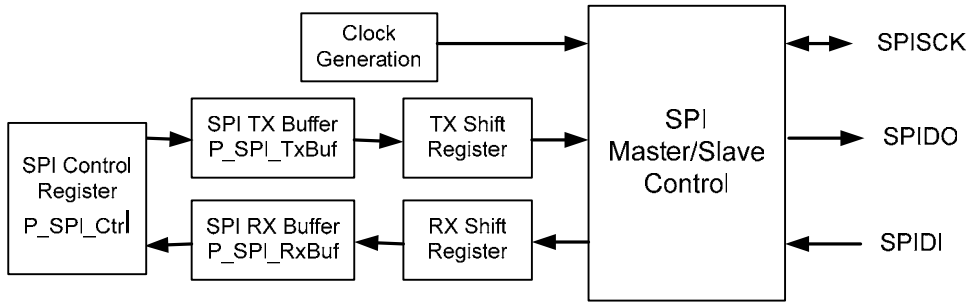


图 16-2 SPI 接口的功能框图

16.2SPI 控制引脚配置

名称	I/O	描述
SPICLK	I/O*	串行外设接口，时钟引脚(与 IOB.11 共享)
SPIDO	O	串行外设接口，数据输出引脚(与 IOB.13 共享)

SPIDI	I	串行外设接口，数据输入引脚(与 IOB.12 共享)

*当 SPI 在主模式下运行时，SPICLK 为输出；当 SPI 在从模式下运行时，SPICLK 为输入

16.3控制寄存器

SPMC75X 系列微控制器内嵌的 SPI 模块共有 5 个控制寄存器，如表 16-1 所示。通过这 5 个控制寄存器可以完成 SPI 模块所有功能的控制。

表 16-1 比较匹配定时器寄存器

地址	寄存器	名称
7140h	P_SPI_Ctrl	SPI 控制寄存器
7141h	P_SPI_TxStatus	SPI 发送状态寄存器
7142h	P_SPI_TxBuf	SPI 发送缓冲寄存器
7142h	P_SPI_RxStatus	SPI 接收状态寄存器
7143h	P_SPI_RxBuf	SPI 接收缓冲寄存器

16.3.1 SPI 控制寄存器

该寄存器用来设置 SPI 的中断、工作模式、时钟极性、数据采样方式和主模式下 SPI 通信速率。

P_SPI_Ctrl ($7140)：SPI 控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R	R	R	W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SPIE	保留			SPIRST	SPISPCLK		SPIMS

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
保留		SPIPHA	SPIPOL	SPISMPS	SPIFS		

第 15 位　　SPIE：SPI 允许,此位设为“1”，端口 B[13:11]用于 SPI 功能接口，不能再用作 GPIO

0 ＝ 禁止

1 ＝ 允许

第 14:12 位　保留

第 11 位　　SPIRST：写入 1 复位。除寄存器设定之外，此时将产生一个脉冲，复位除寄存器设置以外的 SPI 模块。

第 10:9 位　SPISPCLK：采样时钟选择位. 用于实现噪声抑制功能。

00 ＝ 无取样

01 = F_{CK}

10 = $F_{CK}/2$

11 = $F_{CK}/4$

第 8 位 **SPIMS**： SPI 模式选择

0 = 主模式

1 = 从模式

第 7:6 位 保留

第 5 位 **SPIPHA**: SPI 时钟相位。SPI 时钟相位选择。参考 SPI 主模式时序图 16-3。

第 4 位 **SPIPOL**: SPI 时钟极性。SPI 时钟极性选择。参考 SPI 主模式时序图 16-3。

第 3 位 **SPISMPS**：主模式下的 SPI 采样模式选择

0 = 在数据输出时段的中间对输入数据位进行取样

1 = 在数据输出时段的末端对输入数据位进行取样

第 2:0 位 **SPIFS**：主模式时钟频率选择

000 = $F_{CK}/4$

001 = $F_{CK}/8$

010 = $F_{CK}/16$

011 = $F_{CK}/32$

100 = $F_{CK}/64$

1xx = $F_{CK}/128$

16.3.2 SPI 发送状态寄存器

该寄存器是 SPI 发送相关的标志寄存器，包括发送中断的使能、发送完成中断和发送移位寄存器空标志。

P_SPI_TxStatus ($7141)： SPI 发送状态寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R	R	R	R	R
0	0	0	0	0	0	0	0
SPITXIF	SPITXIE	SPITXBF	保留				

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

第 15 位 **SPITXIF**：SPI 发送中断标志

0 = 未发生

1 = 已发生，写入 1 可清除

第 14 位　　SPITXIE：SPI 发送中断使能

0 = 禁止

1 = 使能

第 13 位　　SPITXBF：发送缓冲满标志。当发送缓冲满时，此位被硬件设为 1，当向 P_SPI_TxBuf 寄存器写入的数据装入移位寄存器中时，此位被清零。SPITXBF 被置 1 时，软件不应更新 P_SPI_TxBuf 寄存器，否则新数据将覆盖即将要发送的旧数据。

0 = 发送缓冲为空

1 = 发送缓冲已满

第 12:0 位　　保留

16.3.3 SPI 发送缓冲寄存器

P_SPI_TxBuf ($7142)： SPI 发送缓冲寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SPITXBUF							

第 15:8 位　　保留

第 7:0 位　　SPITXBUF = 写入数据发送到 SPIDO 引脚

16.3.4 SPI 接收状态寄存器

该寄存器是 SPI 接收相关的标志寄存器，包括接收中断的使能、接收完成中断和数据溢出标志。

P_SPI_RxStatus $(7143)： SPI 接收状态寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R	R	R	R/W	R	R
0	0	0	0	0	0	0	0

SPIRXIF	SPIRXIE	保留	FERR	保留

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

第 15 位　SPIRXIF：SPI 接收中断标志

0 ＝ 未发生

1 ＝ 已发生，写入 1 可清除

第 14 位　SPIRXIE：SPI 接收中断允许

0 ＝ 禁止

1 ＝ 允许

第 13:11 位　保留

第 10 位　FERR：数据覆盖错误标志

0 ＝ 未发生覆盖错误

1 ＝ 已发生错误覆盖

第 9:0 位　保留

16.3.5　SPI 接收缓冲寄存器

P_SPI_RxBuf ($7144)：　SPI 接收缓冲寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保存							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SPIRXBUF							

第 15:8 位　保留

第 7:0 位　SPIRXBUF ＝ 从 SPIDI 引脚接收到数据

16.4SPI 运行模式

SPMC75X 系列微控制器内嵌的 SPI 通信模块支持主从两种工作模式。可以根据需要选择合适的工作模式。

16.4.1 主模式

在主模式下，移位时钟(SPICLK)由 SPI 模块产生。在 P_SPI_Ctrl 寄存器中有两位用作对时钟相位(SPIPHA)和极性(SPIPOL) 位的控制。在软件向 P_SPI_TxBuf 寄存器写入一个字节之后，数据被锁存到寄存器的内部发送缓冲中。如果此时移位寄存器没有执行数据移位操作，该数据将被载入到移位寄存器中并在下一个 SPICLK 时钟开始传输。如果移位寄存器正在执行数据移位(由 P_SPI_TxStatus 寄存器中的 SPITXBF 标志得知)，新数据会等待当前的数据移出后再进行移位。

SPI 通过 SDO 引脚将数据从最高有效位(MSB)移到最低有效位(LSB)。8 位数据在 8 个 SCLK 周期后全部移出。同时，接收的数据也通过 SDI 引脚移入。当每组 8 位发送完成后，P_SPI_TxStatus 寄存器中的 SPITXIF 置位；此外，如果 P_SPI_TxStatus 寄存器中的 SPITXIE 位被设置为‘1’，会产生一个 SPI 发送中断。同样，当 SPI 接口成功地接收了一组 8 位字节时，接收到的数据将被锁存到接收缓冲器中。此时，P_SPI_RxStatus 寄存器中的 SPIRXIF 位将被设置为‘1’，并且发生一个 SPI 接收中断。

图 16-3 给出了 SPI 主模式下不同运行类型的时序(极性控制位等于“1”或“0”，相位控制位等于“1”或“0”，取样控制位等于“1”或“0”)

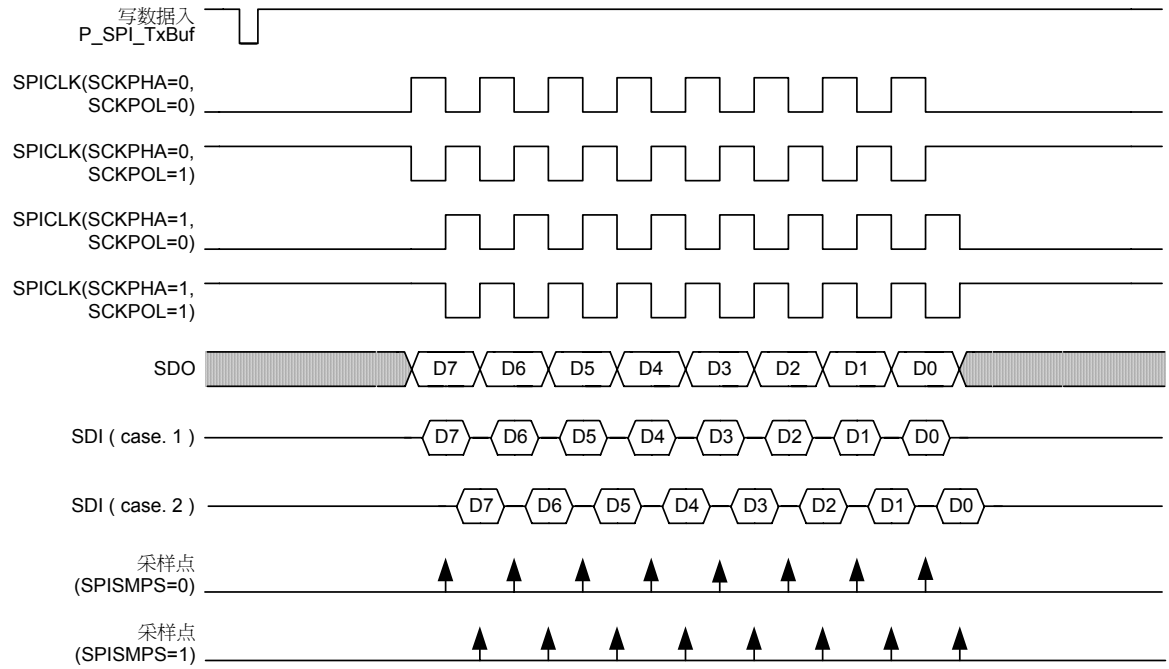


图 16-3 SPI 主模式时序图

16.4.2 从模式

在从模式下，移位时钟 SPICLK 来自外部 SPI 主设备，所以从第一个外部时钟周期开始传输。发送前，软件应在第一个来自主设备的 SPICLK 之前向其发送缓冲写入数据。主设备与从设备都必须按相同的 SPICLK 相位和极性运行，以进行数据的发送与接收。如果时钟相位(SPIPHA)为“1”，只要向 P_SPI_TxBUF 寄存器写入数据，就开始移出第一个数据位。如果时钟相位(SPIPHA)为“0”，则在第一个 SPICLK 边沿后才开始移出第一个数据位。

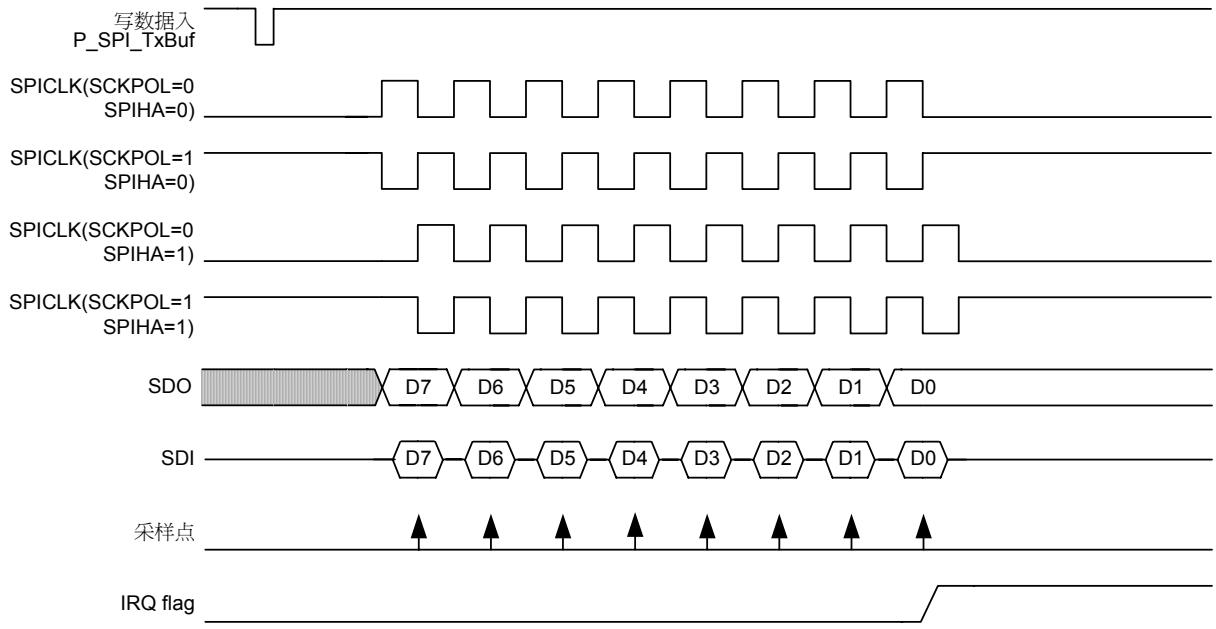


图 16-4　SPI 从模式时序图

16.5设计参考

【例 16-1】设置 SPI 为主模式并发送数据

```
P_SPI_Ctrl->B.SPIRST = CB_SPI_SPIRST;                    /* SPI 复位*/
P_SPI_Ctrl->W = CW_SPI_SPIE + CW_SPI_SPISPCLK_FCKdiv4 +
                CW_SPI_SPIFS_CPUCLKdiv64;                /* SPI 使能 */
Data = 0x0000;
while(1)
{
P_SPI_TxBuf   ->W = Data;
while(P_SPI_TxStatus->B.SPITXBF);
P_SPI_TxStatus->B.SPITXIF = CB_SPI_SPITXIF;              /* 清除 SPI 中断标志 */
F_Delay(10);                                             /* 延时 10.0 ms 用于调试 */
Data++;
}    }                                                   /* 结束 while(1)循环 */
```

例 16-1 SPI 查询方式发送数据

【例 16-2】设置 SPI 为主模式并用 IRQ6 中断发送数据

```
P_SPI_Ctrl ->B.SPIRST = CB_SPI_SPIRST;                   /* SPI 复位 */
P_SPI_Ctrl ->W= CW_SPI_SPIE + CW_SPI_SPISPCLK_FCKdiv4 +
CW_SPI_SPIFS_CPUCLKdiv64;                                /* SPI 使能 */
P_SPI_TxStatus ->B.SPITXIE = CB_SPI_SPITXIE;             /* 使能 SPI 中断 */

IRQ_ON();                                                /* 使能 IRQ */
Data = 0x0000;
P_SPI_TxBuf-> W = Data;
```

```
while(1);

/*********************************************************************************************/
void IRQ6(void) __attribute__ ((ISR));
void IRQ6(void)
{
if(P_SPI_TxStatus->B.SPITXIF)                          /* 查询 SPI 中断标志 */
{
P_SPI_TxStatus->B.SPITXIF = CB_SPI_SPITXIF;            /* 清除 SPI 中断标志 */
Data++;
P_SPI_TxBuf   -> W = Data;
}                                                      /* 结束 if */
}
```

例 16-2　SPI 中断方式发送数据

第17章 通用异步串行通讯-UART

17.1简介

SPMC75X 系列微控制器内置一个 UART 模块，可以完成以下功能：接收数据，将外部设备串行数据进行转换为并行数据；发送数据，将并行数据转换为串行数据，对外发送。此模块具有以下特点。

- 提供标准的异步全双工通讯
- 可编程的收发波特率
- 可进行偶校验、奇校验或禁止校验
- 停止位可设置为 1 位或 2 位
- 支持发送中断
- 支持接收中断
- 高抗噪声能力的数据接收(接收中间连续进行 3 次采样，并对结果进行多数决策)
- 在接收中进行帧校验和奇偶校验
- 溢出侦测
- 波特率可在 1200 bps 到 115200 bps 之间编程设定
- 可选择 TXD1/RXD1 和 TXD2/RXD2 任意一组作为发送/接收数据通道
- 可独立激活发送/接收功能

17.1.1 UART 帧结构

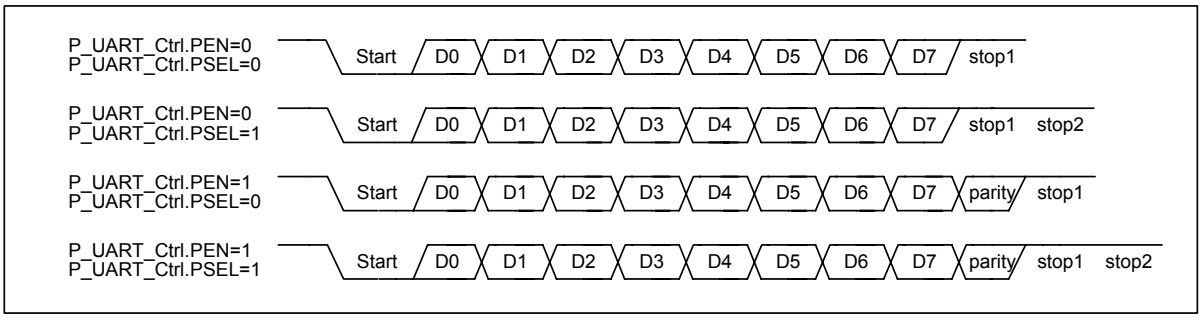


图 17－2 UART 帧数据格式

17.1.2 UART 引脚配置

名称	I/O	描述
RXD1	I	UART 接收引脚(与 IOB12 共享)
TXD1	O	UART 发送引脚(与 IOB13 共享)
RXD2	I	UART 接收引脚(与端口 IOC0 共享)
TXD2	O	UART 发送引脚(与端口 IOC1 共享)

17.2控制寄存器

SPMC75X 系列微控制器内嵌的 UART 模块共有 5 个控制寄存器，如表 17-1 所示。通过这 4 个控制寄存器可以完成 UART 模块所有功能的控制。

表 17-1　UART 模块控制寄存器

地址	寄存器	名称
7100h	P_UART_Data	UART 数据寄存器
7101h	P_UART_RXStatus	UART 错误状态寄存器
7102h	P_UART_Ctrl	UART 控制寄存器
7103h	P_UART_BaudRate	UART 波特率设定寄存器
7104h	P_UART_Status	UART 状态寄存器

17.2.1　UART 数据寄存器

P_UART_Data ($7100)：UART 数据寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留				OE	保留	FE	PE

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
UARTDATA							

第 15:12 位　保留

第 11 位　**OE**：溢出错误（只读）。 与 P_UART_RXStatus 的第 3 位相同，如果在接收数据时因上一次接收的数据没读走而被覆盖时，此位被置 1。

0 ＝ 未发生

1 ＝ 已发生

第 10 位　保留

第 9 位　**PE**：奇偶校验错误（只读）。与 P_UART_RXStatus 的第 1 位相同，如果接收数据字符的奇偶性与在 PSEL 控制位中选择的奇偶性不匹配的话，此位被置“1”。

0 ＝ 未发生

1 ＝ 已发生

第 8 位　**FE**：帧错误(只读)。与 P_UART_RXStatus 的第 0 位相同，如果接收的字符不是一个合法的停止位(一个合法的停止位至少为 1 比特)，此位被置“1”。

0 = 未发生

1 = 已发生

第 7:0 位　　UARTDATA：UART 数据读/写寄存器

此控制寄存器是发送/接收缓冲的入口

注：第[11:8]位的错误标志与 P_UART_RXStatus 寄存器中第[3..0]位相同。

17.2.2 UART 错误状态寄存器

UART 接收状态标志寄存器 P_UART_RXStatus。标识接收出错状态。注意：P_UART_RXStatus 的内容是通过读取 P_UART_Data 更新的。如果在读取 P_UART_Data 之前读取 P_UART_RXStatus，则相应的状态标志不是最新的，而是上一次的值。同时，P_UART_RXStatus 中的状态标志只能通过读取 P_UART_Data 时自动置位或是清除。不能通过软件向相应位写 1 清除。

P_UART_RXStatus ($7101)： UART 错误状态寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
保留				OE	保留	PE	FE

第 15:4 位　保留

第 3 位　　OE：溢出错误（只读）。 与 P_UART_RXStatus 的第 3 位相同，如果在接收数据时因上一次接收的数据没读走而被覆盖时。

读出0 = 未发生

读出1 = 已发生

第 2 位　　保留

第 1 位　　PE：奇偶性错误（只读）。如果接收数据字符的奇偶性与在 PSEL 控制位中选择的奇偶性不匹配的话，此位被置“1”。

读出0 = 未发生

读出1 = 已发生

第 0 位　　FE：帧错误。如果接收的字符不具有一个合法的停止位的话(一个合法的停止位为 1 比特)，此位被置“1”。

读出0 = 未发生

读出1 = 已发生

17.2.3 UART 控制寄存器

P_UART_Ctrl ($7102)： UART 控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R	R/W	W	R/W	R/W	R
0	0	0	0	0	0	0	0
RXIE	TXIE	RXEN	TXEN	Reset	TXCHSEL	RXCHSEL	保留

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R/W	R/W	R/W	R
0	0	0	0	0	0	0	0
保留				SBSEL	PSEL	PEN	保留

第 15 位 **RXIE**：接收中断使能。如果此位置“1”，只要接收缓冲收到数据，硬件会产生中断，中断可选为 IRQ6 或 FIQ。

0 ＝ 禁止

1 ＝ 使能

第 14 位 **TXIE**：发送中断使能。如果此位置“1”，而且发送缓冲为空，硬件会产生中断，中断可选为 IRQ6 或 FIQ。

0 ＝ 禁止

1 ＝ 使能

第 13 位 **RXEN**：UART 接收使能。如果此位置“1”， 则 UART 接收接口使能，RXD1/RXD2 输入通道由 P_UART_Ctrl 寄存器的 RXCHSEL 位编程控制。

0 ＝ 禁止

1 ＝ 使能

第 12 位 **TXEN**：UART 发送使能。如果此位置“1”， 则 UART 发送接口被使能，TXD1/TXD2 输出通道由 P_UART_Ctrl 寄存器的 TXCHSEL 位编程控制。

0 ＝ 禁止

1 ＝ 使能

第 11 位 **Reset**：UART 模块复位控制

0 ＝ 不复位

1 ＝ 复位UART模块

第 10 位 **TXCHSEL**：发送数据通道选择

0 ＝ 在IOC1引脚上向TXD2串行数据发送

1 ＝ 在IOB13引脚上向TXD1的串行发送

第 9 位 **RXCHSEL**：接收数据通道选择

0 = 在IOC0引脚上接收来自RXD2的串行数据

1 = 在IOB12引脚上接受来自RXD1的串行数据

第 8:4 位　保留

第 3 位　SBSEL：停止位位数选择。当此位被置 “1”时，发送帧末端会发送两个停止位。

0 = 1个停止位

1 = 2个停止位

第 2 位　PSEL：奇偶校验选择。如果此位被置 “1”，则在发送与接收中执行偶校验，即检查在数据以及奇偶位中 1 的个数是否为偶数。当此位被清零时，则执行奇校验，即检查 1 的个数是否为奇数。当 PEN 控制位禁止奇偶校验位时，此位无效。

0 = 奇校验（如果 PEN= 1）

1 = 偶校验（如果 PEN= 1）

第 1 位　PEN：奇偶校验使能位。如果此位被置 “1”，则使能奇偶校验，否则禁止，数据帧中不加入奇偶校验位。

0 = 禁止

1 = 使能

第 0 位　保留

17.2.4 UART 波特率设定寄存器

P_UART_BaudRate($7103)：UART 波特率设定寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
UARTBUD							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
UARTBUD							

第 15:0 位　UARTBUD：UART 波特率

Baud Rate = F_{CK} / [16 x (65536 - P_UART_BaudRate)]

依据上式，可得出下面计算P_UART_BaudRate寄存器的值的公式P_UART_BaudRate = 65536 - F_{CK} / (16 x Baud Rate)

17.2.5 UART 状态寄存器

P_UART_Status ($7104)： UART 状态寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	1	0	0	0	0	0	0
RXIF	TXIF	保留					

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留	RXBF	保留		BY	保留		

第 15 位 RXIF：接收中断标志。此标志指示出一个新数据接收完毕并发出 UART 接收中断。当数据从 P_UART_Data 寄存器中读出或由 RXEN 置’0’禁止接收时，RXIF 位自动清零。

0 ＝ 无接收中断

1 ＝ 完成一个字节的接收，如果RXIE位置‘1’，则产生一个中断

第 14 位 TXIF：发送中断标志。如果发送缓冲就绪，则此标志被硬件置‘1’，而且发出 UART 发送中断。当新数据写入 P_UART_Data 寄存器或由 TXEN 置’0’禁止发送时，TXIF 位自动清零。

0 ＝ 发送器未准备好

1 ＝ 发送器准备好，如果TXIE位被置‘1’，则产生一个中断

第 13:7 位 保留

第 6 位 RXBF：接收缓冲已满标志。此标志为只读，硬件会自动设定或清除此标志。

0 ＝ 接收缓冲未满

1 ＝ 接收缓冲已满

第 5:4 位 保留

第 3 位 BY：发送器忙标志。如果此位读出的值为“1”，则说明 UART 模块正在发送数据。直到整个字节(包括所有的停止位)都从移位寄存器被发送出后，此位才被清零。注意此标志为只读，硬件会自动设定或清除此标志。

0 ＝ 发送器准备好

1 ＝ 发送器忙

第 2:0 位 保留

17.3UART 的操作

17.3.1 波特率设置

SPMC75X 系列微控制器的波特率是由一个 16 位专用定时器产生的。定时器每次从其设置值(最大 0xFFFF)递增计数,当值为 0xFFFF 定时器自动重新载入波特率寄存器中的值,产生一个时钟。实际的波特率为这个时钟的 16 分频。

$$\text{Baud Rate} = F_{CK} / [\ 16 \times (65536 - \text{P_UART_BaudRate})]$$

波特率寄存器中的内容为 16 位的无符号数。使用下面的公式可从一个已知的波特率导出所需的波特率寄存器值：

$$\text{P_UART_BaudRate} = 65536 - F_{CK} / (16 \times \text{Baud Rate})$$

表 17-2 F_{CK} = 24 MHz 时的 P_UART_BaudRate 的设定值

波特率	波特率定时器重装寄存器值@ 24MHz
115200 bps	0xFFF3
57600 bps	0xFFE6
19200 bps	0xFFB2
9600 bps	0xFF64
4800 bps	0xFEC8
2400 bps	0xFD8F
1200 bps	0xFB1E

17.3.2 数据发送

软件向 P_UART_Data 寄存器写入数据后，在第一个时钟沿时,UART 开始发送。UART 按照以下的顺序在 TXD2/TXD1 引脚上发送数据：开始位、8 个数据位(LSB 在前，MSB 在后)、奇偶位(只在使能奇偶模式下)、停止位。当发送停止位时，P_UART_Status 寄存器中的 TXIF 在停止位发送后的两个 FCK 周期后被置位。当新数据写入 P_UART_Data 寄存器时，TXIF 位清零。发送时序如图 17-1 所示。

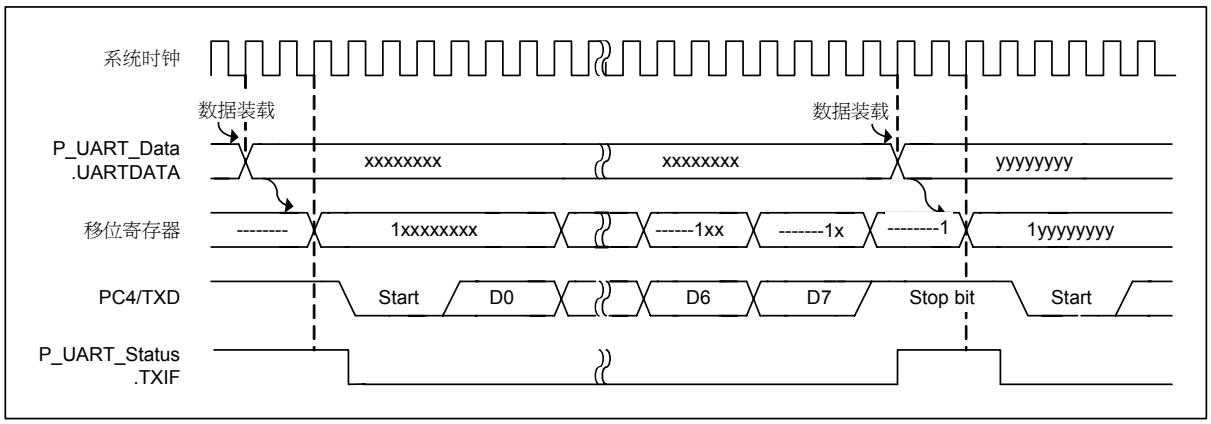


图 17-1 UART 发送时序图

17.3.3 数据接收

UART 的接收使能后，接收过程从接收到一个起始位开始。当检测到有效的起始位后，16 分频时钟作为波特率计数时钟，复位后与接收位边沿对齐，开始接收数据。为抑制噪声，串口会在每位中部进行 3 次连续采样，通过多数决议来确定每个接收位的值。如果多数表决发生在起始位检测，只要不是连续的三次低电平，则当前帧会被丢弃。接收模块则停止当前接收过程并等待下一个数据帧。数据采样方式如图 17-2 所示。

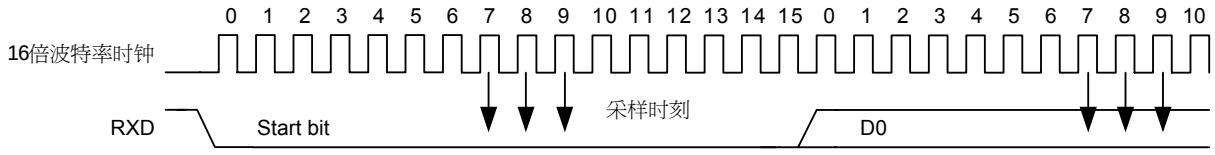


图 17-2 数据接收采样方式

接收到停止位后,UART 模块会把数据传到 P_UART_Data 寄存器中并置 RXIF 和 RXBF。接收模块则等待接收下一个数据帧,接收时序如图 17-3 所示。

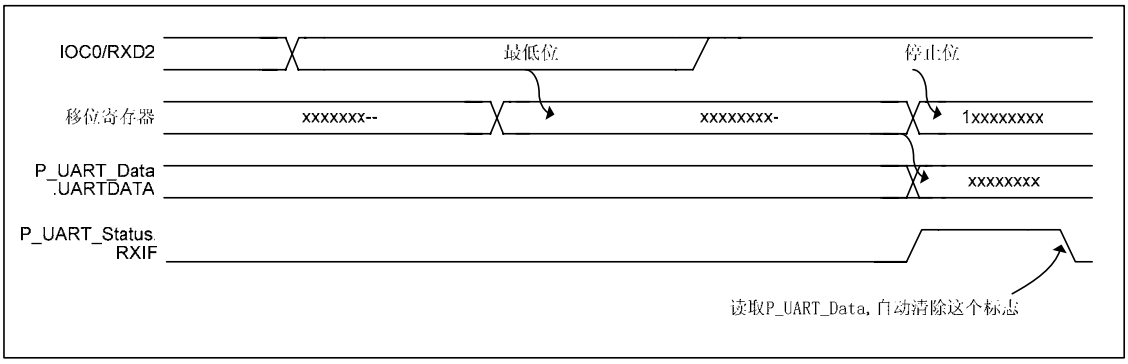


图 17-3 UART 接收时序

如果在新的接收数据完成时,原接收数据没有被读出,则新的数据就会覆盖旧数据，同时会置溢出标志 OE。 注意：OE 会在下一次校验通后自动清零。数据溢出事件时序如图 17-4 所示。

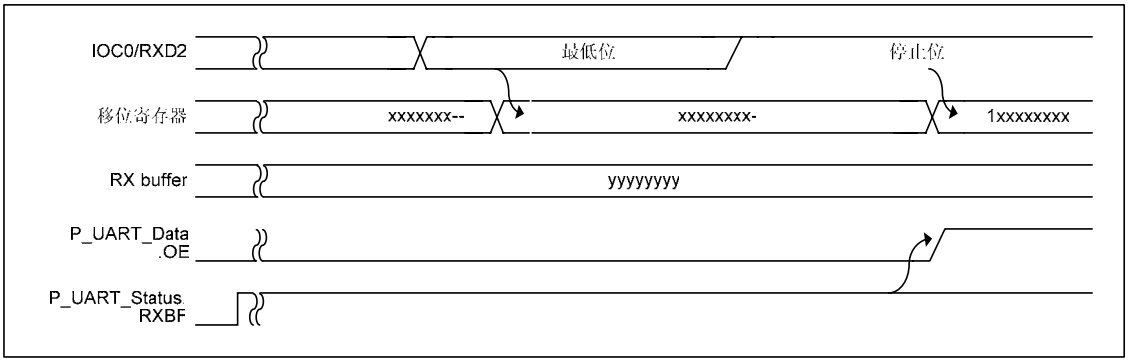


图17-4 接收数据溢出 OE 时序

SPMC75X 系列微控制器的 UART 支持奇偶校验，通过设置 P_UART_Ctrl.PSEL 来选择到底是

偶校验还是奇校验。如果 P_UART_Ctrl.PEN 置位,则接收完奇偶校验位就执行奇偶校验。如果奇偶校验结果出错，则会将 PE 位置 1。相应的时序请参考图 17-5 和图 17-6。如果数据接收的帧不完整，则 FE 将被置 1。 注意：若在下一次校验通过，FE 将自动清零。

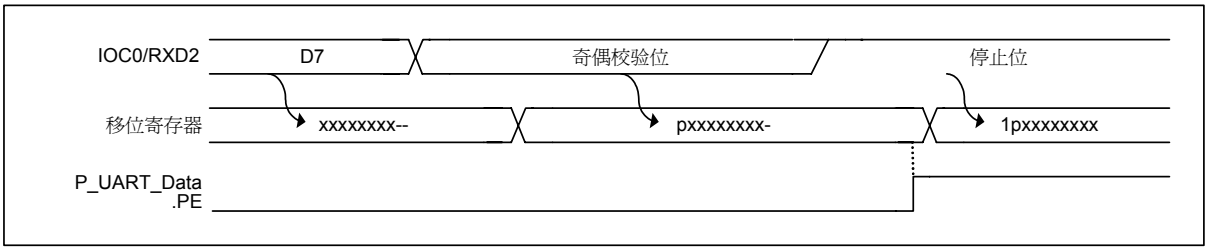


图17-5 奇偶出错 FE 时序

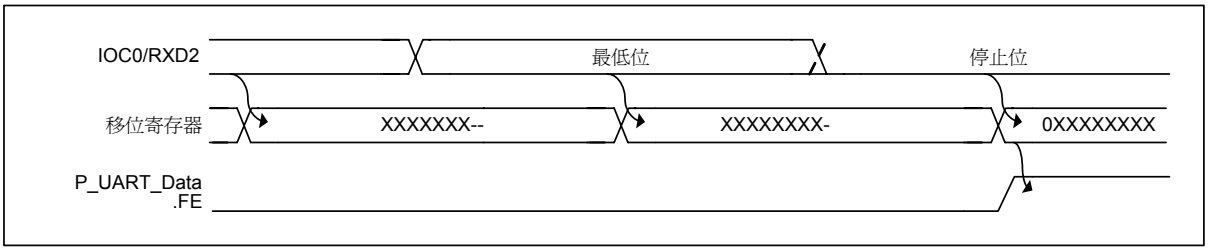


图17-6 帧出错 FE 时序

17.4设计参考

【例 17-1】全双工模式，UART 使用 RXD2 与 TXD2 作为通信引脚

```
void UART_Test(void)
{
unsigned int code = 0;

P_UART_Ctrl->W = CW_UART_RXCHSEL_No2 | CW_UART_TXCHSEL_No2 | \
CW_UART_RXEN | CW_UART_TXEN | CW_UART_RXIE | \
CW_UART_TXTE;
/* 收发引脚 RXD2/TXD2 使能, UART 使能, 使能 RX 中断*/
P_UART_BaudRate->W = 65406;                    /* 9600 bps ,cpu 时钟 20 MHz */
P_UART_RXStatus->W = 0x000B;                   /* 清除 OE, FE, PE 错误标志 */
}                                              /* UART_Test() */

/*****************************************************************************/
unsigned int txd_data = 0;
void IRQ6(void) __attribute__ ((ISR));
void IRQ6(void)
{
unsigned int msg;
if(P_INT_Status->B.UARTIF)
```

```
{
if(P_UART_Status->B.TXIF && P_UART_Ctrl->B.TXIE)
{
// transmit data buffer to TXD2
P_UART_Data->W = txd_data;                          /* 发送数据 */
}
if(P_UART_Status->B.RXIF)
{
// receive data from RXD2
msg = P_UART_Data->W;                               /* 接收数据 */
}
}
}                                                    /* IRQ6() */
```

例 17-1　UART 中断设置

第18章 10 位模/数转换器

18.1简介

SPMC75X 系列微控制器内嵌一个 100Ksps 转换速率的高性能 10 位通用 ADC 模块，采用 SAR（逐次逼近）结构。它与 IOA[7..0]复用引脚作为输入通道，最多能提供 8 路模拟输入能力。同时，ADC 模块有多种工作模式可选，它的转换触发信号可以是软件产生也可以通过来自外部（IOA15）、PDC 位置侦测、MCP 等定时器的信号，以满足不同的应用。

利用此 ADC 模块，可以同电机驱动定时器联合动作，实现电机驱动过程中电参量的同步测量，满足电机驱动的需要。此外，ADC 模块也可以实现一些普通的模拟测量动作，如电压测量、温度信号测量、低频信号的采集等。ADC 模块的结构如图 18-1 所示。

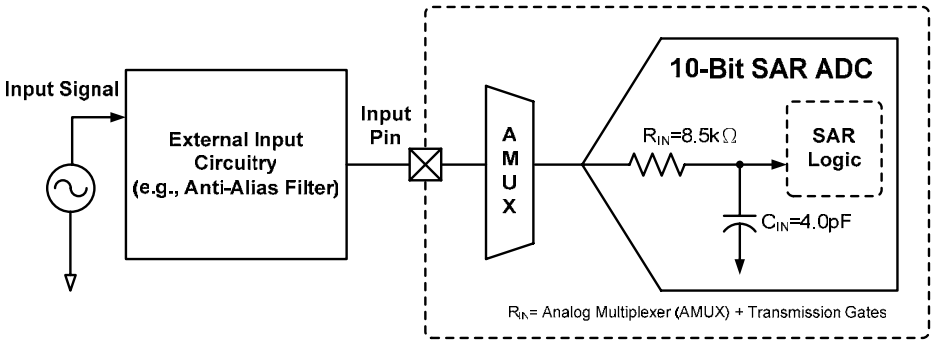


图 18-1　ADC 功能框图

18.2控制寄存器

SPMC75X 系列微控制器内嵌的 ADC 模块共有 4 个控制寄存器，如表 18-1 所示。通过这 4 个控制寄存器可以完成 ADC 模块所有功能的控制。

表 18-1 比较匹配定时器寄存器

地址	寄存器	名称
7160h	P_ADC_Setup	ADC 设置寄存器
7161h	P_ADC_Ctrl	ADC 控制寄存器
7166h	P_ADC_Channel	ADC 输入通道使能寄存器
7162h	P_ADC_Data	ADC 数据寄存器

18.2.1　ADC 设置寄存器

P_ADC_Setup ($7160)：ADC 设置寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
1	0	0	0	0	0	0	0
ADCCS	ADCEN	保留			ADCFS		ADCEXTRG

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
ASPEN	保留						

第 15 位　　ADCCS：ADC 功能模块电路供电

0 ＝ 不为 ADC 功能模块供电

1 ＝ 为 ADC 功能模块供电

第 14 位　　ADCEN：A/D 转换使能，当 A/D 转换工作时，功率消耗将快速增长。所以，只在需要时才使能 A/D 转换，在成功得到 A/D 转换的数据后立即关闭 ADC 通道功能。

0 ＝ 禁止 ADC 模块

1 ＝ 使能 ADC 模块

第 13:11 位　保留

第 10:9 位　　ADCFS：模/数转换器时钟选择，建议 AD 转换器时钟不要超过 1.5MHz。

00 ＝ $F_{CK}/8$

01 ＝ $F_{CK}/16$

10 ＝ $F_{CK}/32$

11 ＝ $F_{CK}/64$

第 8 位　　ADCEXTRG：通过端口 IOA15 触发 A/D 转换。IOA15 可设置为悬浮、上拉和下拉式输入

0 ＝ 禁止

1 ＝ 使能

第 7 位　　ASPEN：自动采样模式使能，采样请求时钟来自定时器/PWM 模块

0 ＝ 禁止

1 ＝ 使能

第 6:0 位　　保留

18.2.2　ADC 控制寄存器

P_ADC_Ctrl($7161)：　ADC 控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R	R	R	R	R	R
0	0	0	0	0	0	0	0
ADCIF	ADCIE	保留					

B7	B6	B5	B4	B3	B2	B1	B0
R	R/W	R	R	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
ADCRDY	ADCSTR	保留			ADCCHS		

第 15 位　　ADCIF：ADC 中断标志。表示 ADC 转换完成的状态标志。写入‘1’ 可清除此标志。

0 = 未发生中断

1 = 发生中断

第 14 位　　ADCIE：使能 ADC 中断。使能或禁止 ADC 转换中断的产生。

0 = 禁止

1 = 使能

第 13：8 位　保留

第 7 位　　ADCRDY：ADC 转换完成。

0 = 转换未完成， AD 转换结果无效

1 = 转换完成，AD 转换结果有效

第 6 位　　ADCSTR：手动启动 AD 转换。向此位写入 “1” 则启动 AD 转换。

0 = 无效

1 = 启动转换

第 5:3 位　保留

第 2:0 位　ADCCHS：选择 ADC 转换器通道输入

000 = ADC 通道 0 (IOA0)

001 = ADC 通道 1 (IOA1)

010 = ADC 通道 2 (IOA2)

011 = ADC 通道 3 (IOA3)

100 = ADC 通道 4 (IOA4)

101 = ADC 通道 5 (IOA5)

110 = ADC 通道 6 (IOA6)

111 = ADC 通道 7 (IOA7)

18.2.3 ADC 输入通道使能寄存器

P_ADC_Channel 寄存器用来使能 IOA[7..0]相应引脚的 ADC 模拟输入功能。需要注意的是：当 IOA[7..0]的相应引脚用用 ADC 模拟输入时，相应的 I/O 功能应设为悬浮输入状态。

P_ADC_Channel($7166H) ： ADC 输入通道使能寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
ADCCH7	ADCCH6	ADCCH5	ADCCH4	ADCCH3	ADCCH2	ADCCH1	ADCCH0

第 15：8 位　保留

第 7 位　　ADCCH7：ADC 输入通道 7 使能

1 ＝ IOA7 为 ADC 通道 7

0 ＝ IOA7 为 GPIO

第 6 位　　ADCCH6：使能 ADC 输入通道 6

1 ＝ IOA6 为 ADC 通道 6

0 ＝ IOA6 为 GPIO

第 5 位　　ADCCH5：使能 ADC 输入通道 5

1 ＝ IOA5 为 ADC 通道 5

0 ＝ IOA5 为 GPIO

第 4 位　　ADCCH4：使能 ADC 输入通道 4

1 ＝ IOA4 为 ADC 通道 4

0 ＝ IOA4 为 GPIO

第 3 位　　ADCCH3：使能 ADC 输入通道 3

1 ＝ IOA3 为 ADC 通道 3

0 ＝ IOA3 为 GPIO

第 2 位　　ADCCH2：使能 ADC 输入通道 2

1 ＝ IOA2 为 ADC 通道 2

0 ＝ IOA2 为 GPIO

第 1 位　　ADCCH1：使能 ADC 输入通道 1

1 ＝ IOA1 为 ADC 通道 1

0 ＝ IOA1 为 GPIO

第 0 位　　ADCCH0：使能 ADC 输入通道 0

1 ＝ IOA0 为 ADC 通道 0

0 ＝ IOA0 为 GPIO

18.2.4　ADC 数据寄存器

P_ADC_Data ($7162H) ：　ADC 数据寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
ADCDATA							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
ADCDATA		保留					

第 15:6 位　**ADDATA**：ADC 转换数据。转换完成后（ADCRDY 置 1），AD 结果在此寄存器高 10 位。

第 5:0 位　保留

18.3 ADC 转换时序

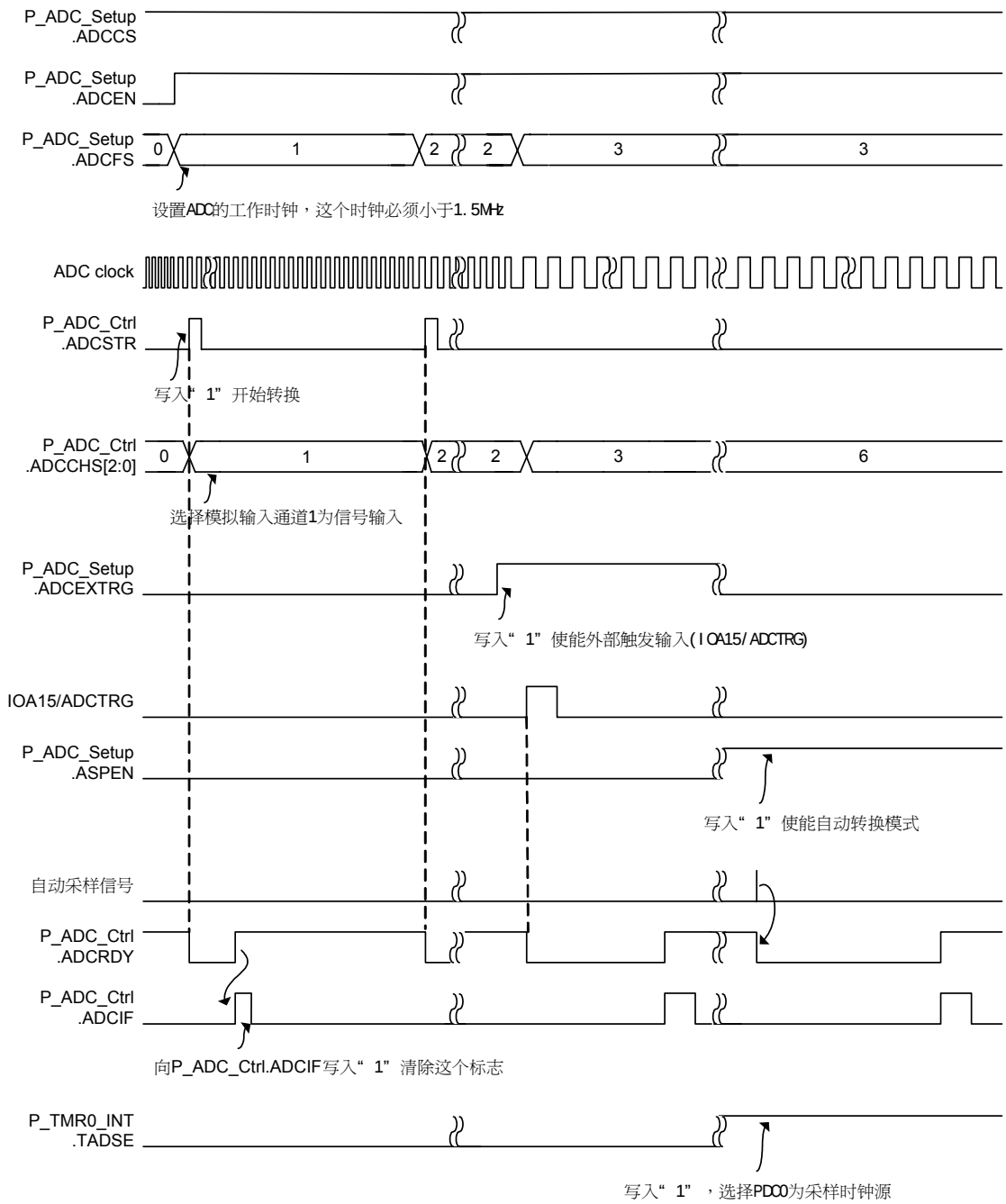


图 18-2　ADC 转化时序图

1. A：Ts1：A/D 转换使能建立时间；min. 2ms @CL=0.1uF for VCM.
2. B：Ts2：改变 A/D 转换通道建立时间；min. 7us @CL=1000pF 4 for PA[7:0] ADC channel loading。Ts2 与 A/D 转换通道输入成比例
3. C：Ts3：外部 A/D 转换的最高电压的建立时间；min. 2us（需首先设定 VRT 电压）
4. D：最大采样率为 100KHz

18.4使用注意事项

ADC 设置寄存器 P_ADC_Setup 控制 ADC 模块上电/掉电, ADC 转换时钟和启动源选择。需要注意的是：上电复位后,ADC 模块掉电、ADC 功能被禁止。这期间, ADC 数据寄存器 P_ADC_Data = 0xFFC0 、 ADC 就绪信号“ADCRDY”为“0”。 如果这时设置 ADC 模块上电并使能 ADC 模块（ADCEN = 1）, 那么 ADC 模块将产生就绪信号 ADCRDY ，同时设置 P_ADC_Ctrl 寄存器的 ADCIF 为”1”。因此， ADC 模块在使能后的第一个转换数据是无效的，请不要使用这一个转换值。

SPMC75X 系列微控制器的 ADC 模块的工作时钟 < 1.5 MHz。请不要将 ADC 模块的工作时钟设置在 1.5 MHz 以上, 否则 ADC 的转换结果将是不确定的。

18.5设计参考

【例 18-1】查询方式 A/D 转换

```
    unsigned int Times;
    unsigned int uiAdcData[8] = {0};                          /* ADC 结果缓冲器 */

    void ADConvert（）
    {
    P_ADC_Setup->W = CW_ADC_ADCCS_Select + CW_ADC_ADCEN +
CW_ADC_ADCFS_CPUCLKdiv64;
    /* AD 转换使能;A/D 转换器时钟选择： CPUCLK/64 */
    P_ADC_Channel->W = 0x00FF;                                /* AD 转换输入通道 7～0 使能 */

    for(i = 0; i < 8; i++)
    {
    P_ADC_Ctrl->B.ADCCHS = i;                                 /* 设置下一个 AD 转换通道*/
    P_ADC_Ctrl->B.ADCSTR = CB_ADC_ADCSTR;                     /* 手动启动 AD 转换 */
    while(!(P_ADC_Ctrl->B.ADCRDY & CB_ADC_ADCRDY));/* 查询 AD 转换是否完成 */
    uiAdcData[i] = (P_ADC_Data->W >> 6);                      /* 高 10 位有效，移到低位 */
    }
     P_ADC_Setup->B.ADCEN = ～CB_ADC_ADCEN;
     }
```

例 18-1　查询方式 A/D 转换

【例 18-2】中断方式 A/D 转换

```
    unsigned int Times;
    unsigned int uiAdcData[8] = {0};              /* ADC 结果缓冲器 */

    void ADConvert（）
    {
    P_ADC_Setup->W = CW_ADC_ADCCS_Select+CW_ADC_ADCEN +
CW_ADC_ADCFS_CPUCLKdiv64;
```

```
                                        /* AD 转换使能;A/D 转换时钟选择
                                        CPUCLK/64 */
P_ADC_Channel->W = 0x00FF;              /* AD 输入通道 7～0 使能 */
Times       = 0x0000;
P_ADC_Ctrl->W = CW_ADC_ADCIE + CW_ADC_ADCSTR + CW_ADC_ADCCHS_Ch0;
                                    /*使能中断，手动启动 AD 转换，从通道 0 开始*/
INT_IRQ();

while(Times<8);
P_ADC_Ctrl->B.ADCIE =  ～CW_ADC_ADCIE;//禁止 AD 转换中断
P_ADC_Setup->B.ADCEN =  ～CB_ADC_ADCCS_Select;
}



/*********************************************************************/
extern unsigned int Times;
extern unsigned int uiAdcData[8];

void IRQ7(void) __attribute__ ((ISR));
void IRQ7(void)
{
if(P_ADC_Ctrl->B.ADCIF & CB_ADC_ADCIF)  /* 查询 AD 转换中断标志 */
{
P_ADC_Ctrl->B.ADCIF     = CB_ADC_ADCIF; /* 清除 AD 转换中断标志 */
uiAdcData[Times] = (P_ADC_Data->W >> 6);  /* 高 10 位有效，移到低位 */

Times++;
P_ADC_Ctrl->B.ADCCHS = Times;                /* 设置下一个 AD 转换通道 */
P_ADC_Ctrl->B.ADCSTR = CB_ADC_ADCSTR;/* 手动启动 AD 转换 */
}
}
```

例 18-2　中断方式 A/D 转换

第19章 看门狗

19.1简介

看门狗定时器的主要功能是：当程序发生问题时使系统重启动。如果 WDT（看门狗定时器）超过了设定时间，则发生系统或是 CPU 复位（通设设置决定）。如果不想使用 WDT 功能，可以在 P_System_Option（0x8000）中禁止该功能。

SPMC75X 系列微控制器的 WDT 是一个 8 位定时器，可选择 8 种不同的时钟源。其工作时序如图 19-1 所示。

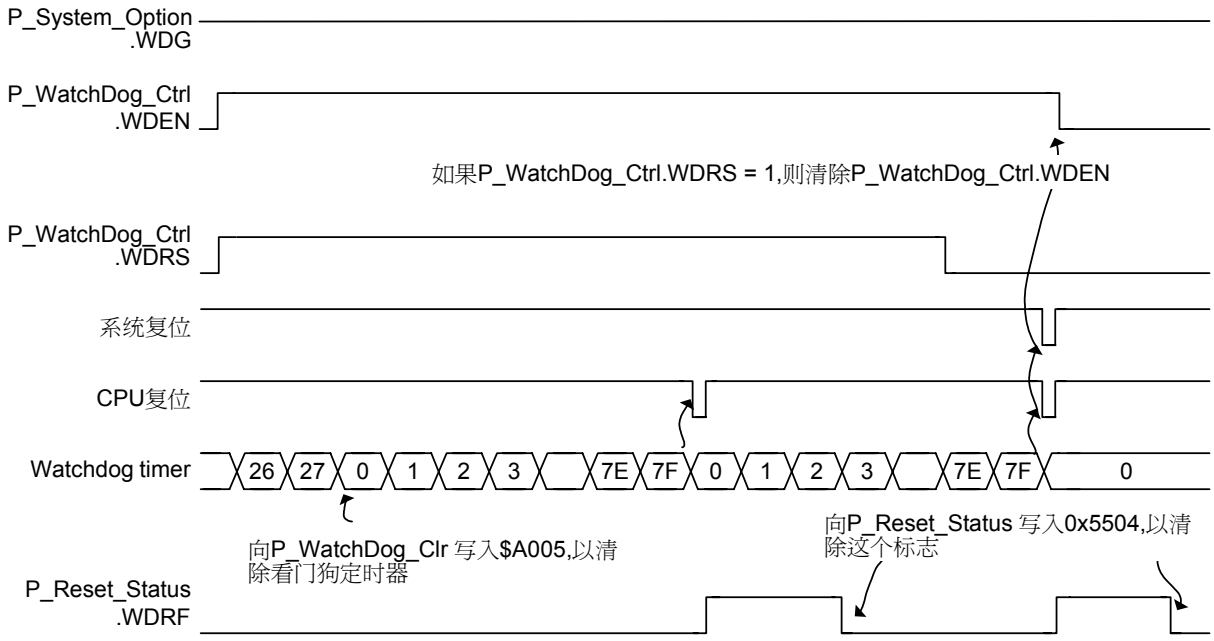


图 19-1 看门狗定时器时序

19.2控制寄存器

SPMC75X 系列微控制器内嵌的看门狗定时器(WDT)模块共有 2 个控制寄存器，如表 19-1 所示。通过这 2 个控制寄存器可以完成看门狗定时器模块所有功能的控制。

表 19-1 看门狗定时器(WDT)寄存器

地址	寄存器	名称
700Ah	P_WatchDog_Ctrl	看门狗控制寄存器
700Bh	P_WatchDog_Clr	看门狗清除寄存器

19.2.1 看门狗控制寄存器

P_WatchDog_Ctrl 此寄存器向看门狗提供了清空定时器和软件设定的开/关功能

P_WatchDog_Ctrl ($700A)：看门狗控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R	R	R	R	R	R
0	0	0	0	0	0	0	0
WDEN	WDRS	保留					

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
WDCHK					WDPS		

第 15 位　**WDEN**：看门狗定时器使能位。此位可以使能/禁止看门狗定时器。一旦使能，看门狗定时器不能被软件禁止。当系统复位时，此位将被复位到默认值。

0 ＝ 禁止

1 ＝ 使能

第 14 位　**WDRS**：看门狗复位选择位

此位用于选择当看门狗定时器溢出时，是系统还是 CPU 被复位

0 ＝ 系统复位

1 ＝ CPU 复位

第 13:8 位　保留

第 7:3 位　**WDCHK**：看门狗控制寄存器检查位。要更改 P_WatchDog_Ctrl 的设定，必须向这些位写入“10101”，否则会产生看门狗复位。这些位的读出值将为 0。

第 2:0 位　**WDPS**：看门狗定时器超时选择，如表 19-2 所示

表 19-2　WDT 定时时间表

WDPS	WDT时钟率(Hz)	超时时间(FCK=24MHz)
000	F_{CK}/65536	699.05ms
001	F_{CK}/32768	349.52ms
010	F_{CK}/16384	174.76ms
011	F_{CK}/8192	87.38ms
100	F_{CK}/4096	43.69ms
101	F_{CK}/2048	21.84ms
110	F_{CK}/1024	10.92ms
111	F_{CK}/512	5.46ms

19.2.2　看门狗清除寄存器

P_WatchDog_Clr 寄存器用于清除看门狗定时器，写入 0xA005 可以清除看门狗定时器。如果写入其它值，将产生看门狗复位，P_Reset_Status 的 WDRF 被置 1。

P_WatchDog_Clr ($700B)：看门狗清除寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
WDTCLR							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
WDTCLR							

19.3设计参考

【例 19-1】看门狗定时器复位@ FCK / 32768

```
    P_WatchDog_Ctrl->W = CW_WDEN + CW_WDRS_CPU_Reset + CW_WDCHK_Setting +
CW_WDPS_FCKdiv32768;
                                                        /* 使能看门狗 */
    while(1)
    P_WatchDog_Clr->W = CW_WatchDog_Clear;              /* 清除看门狗定时器 */
```

例 19-1 看门狗定时器设定

第20章 指令集

20.1简介

SPMC75X 系例微控制器使用凌阳创新科技公司的μ'nSP™内核。μ'nSP™是一种高效的 16 位 RISC 内核，共有 33 条指令，支持立即数寻址、直接寻址、寄存器寻址、寄存器间接寻址、基址变址引址和多重间接寻址 6 种寻址方式。SPMC75X 系列微控制器的指令类型、操作方式和执行周期如表 20-1 所示。表中内容的含义是：

1. **RW** 存贮器读取等待周期，如果无等待周期插入，RW=1，如果等待周期为 N，RW = N;
2. **SW** 存贮器写入等待周期，SW= 0～N;
3. **SRW** 存贮器写入或读取等待周期，如果 ALU 执行存贮操作 SRW = SW，否则 SRW = RW。

表 20-1 指令类型

类型	操作	周期
JMPF	Goto label	5+2RW
DSI6	DS=I6	2+RW
JMPR	Goto MR	4+RW
CALL	CALL label	9+2RW+2SW
FIR_MOV	FIR_MOV_ON/OFF	2+RW
Fraction	Fraction ON/OFF	2+RW
INT SET	INT FIQ/IRQ	2+RW
IRQ	IRQ ON/OFF	2+RW
SECBANK	SECBANK ON/OFF	2+RW
FIQ	FIQ ON/OFF	2+RW
IRQ Nest Mode	IRQNEST ON/OFF	2+RW
BREAK	BREAK	10+2RW+2SW
CALLR	CALL MR	8+RW+2SW
DIVS	DIVS MR,R2	2+RW
DIVQ	DIVQ MR,R2	3+RW
EXP	R2= EXP R4	2+RW
NOP	NOP	2+RW
DS Access	DS=Rs/ Rs=Ds	2+RW
FR Access	FR=Rs/ Rs=FR	2+RW
MUL	MR = Rd* Rs,{ss,us,uu}	12+RW / 13+RW (uu)
MULS	MR = [Rd]*[Rs], size,{ss,us,uu}	us,ss : 10*N+6 + (N+1)*2*RW + {N*SW} / uu: 11*N+6 + (N+1)*2*RW + {N*SW}
Register BITOP	BITOP Rd,Rs	4+RW
Register BITOP	BITOP Rd,offset	4+RW
Memory BITOP	BITOP DS: [Rd],offset	7+2RW+SW
Memory BITOP	BITOP DS: [Rd],Rs	7+2RW+SW
Shift	Rd=Rd LSFT Rs	8+RW
RETI	RETI	8+3RW / 10+4RW (IRQ NEST ON)

类型	操作	周期
RETF	RETF	8+3RW
Base+Disp6	Rd = Rd op [BP+IM6]	6+2RW
Imm6	Rd = Rd op IM6	2+RW
Branch	Jxx label	2+RW / 4+RW (taken)
Indirect	Push/Pop Rx,Ry to [Rs]	4+ 2N + (N+1)RW
DS_Indirect	Rd = Rd op DS: [Rs++]	6+RW+SRW / 7+RW+SRW (PC)
Imm16	Rd = Rs op IMM16	4+2RW / 5+2RW (PC)
Direct16	Rd = Rs op A16	7+2RW+SRW / 8+2RW+SRW (PC)
Direct6	Rd = Rd op A6	5+RW+SRW / 6+RW+SRW (PC)
Register	Rd = Rd op Rs SFT sfc	3+RW / 5+RW (PC)

MULS 周期:10*N+6 + (N+1)*2*RW + {N*SW} (有符号数 x 有符号数, 无符号数 x 有符号数),
11*N+6 + (N+1)*2*RW + {N*SW} (无符号数 x 无符号数), N=1..16, 如果 FIR_MOVE OFF 则(N*SW) = 0

DS_Indirect 周期: 6 + RW + SRW / 7 + RW + SRW (写入 PC)

Direct16 周期: 7 + 2*RW + SRW / 8 + 2*RW + SRW (写入 PC)

Direct6 周期: 5+ RW +SRW / 6+ RW +SRW (写入 PC)

RW: 存储器读周期, RW= 0 ～ N

SW: 存储器写等待周期, SW= 0 ～N. SRW: 存储或读取等待周期

SRW = SW when ALU = store else SRW = RW.

D: 0 (向前跳转) / 1 (向后跳转)

W: 0 (不存储) / 1 (存储)

DS: 0 (不用 DS) / 1 (用 DS)

20.2算术逻辑单元(ALU)操作

表 20-2 指出了 ALU 操作对状态标志的影响。 这四个标志是： N、Z、S、C

N：负数标志

Z：零标志

S：符号位

C：进位标志

表 20-2 操作与标志之间的关系

操作类型	操作	N	Z	S	C	举例
加	a + b	√	√	√	√	Rd=Rd + Rs
带进位的加法	a + b + c	√	√	√	√	Rd=Rd + Rs,c
减	a +～ b + 1	√	√	√	√	Rd=Rd - Rs
带借位的减法	a +～ b + c	√	√	√	√	Rd=Rd - Rs,c

操作类型	操作	N	Z	S	C	举例
比较	a +～ b + 1	√	√	√	√	Cmp Rd,Rs
取补运算	～ b + 1	√	√	-	-	Rd= ～Rs
异或运算	a xor b	√	√	-	-	Rd=Rd ^ Rs
装载操作	a = b	√	√	-	-	Rd=Rs
或操作	a or b	√	√	-	-	Rd=Rd | Rs
与操作	a and b	√	√	-	-	Rd=Rd & Rs
测试操作	Test a, b	√	√	-	-	Test Rd, Rs
存储操作	Store	-	-	-	-	[Rd]=Rs

20.3 条件分支跳转

表 20-3 列出了分支条件。条件匹配必须参考 SR(状态寄存器)。共有 4 类分支标志：N (负值)、Z(零)、S(符号)和 C(进位)。当用户使用一些操作来区别程序流时，N、Z、S、C 标志将被改变。程序员可以使用下面的指令来使程序跳转到匹配地址。表 20-3 给出了程序跳转指令。

表 20-3 程序跳转指令

语法	描述	跳转条件
JCC	进位位为 0	C==0
JB	无符号数小于	C==0
JNAE	无符号数小于	C==0
JCS	进位为 1	C==1
JNB	无符号数不小于	C==1
JAE	无符号数大于等于	C==1
JSC	符号位为 0	S==0
JGE	有符号数大于等于	S==0
JNL	有符号数不小于	S==0
JSS	符号位为 1	S==1
JNGE	有符号数小于	S==1
JL	有符号数小于	S==1
JNE	不相等	Z==0
JNZ	不为零	Z==0
JZ	为零	Z==1
JE	相等	Z==1
JPL	为正数	N==0
JMI	为负数	N==1
JBE	无符号数小于等于	Not (Z==0 and C==1)
JNA	无符号数不大于	Not (Z==0 and C==1)
JNBE	无符号数大于	Z==0 and C==1
JA	无符号数大于	Z==0 and C==1
JLE	有符号数小于等于	Not (Z==0 and S==0)
JNG	有符号数小于等于	Not (Z==0 and S==0)
JNLE	有符号数大于	Z==0 and S==0

语法	描述	跳转条件
JG	有符号数大于	Z==0 and S==0
JVC	未溢出(有符号)	N == S
JVS	溢出(有符号)	N != S
JMP	无条件分支	Always

20.4指令集

N	Z	S	C
-	-	-	-

【 注意 】 “-”为不影响标志; “√” 为影响标志

20.4.1 段长调用

N	Z	S	C
-	-	-	-

【 描 述 】长调用指令，调用地址在代码中给出；PC和SR自动压入堆栈保护。

【 指令周期 】9 + 2*RW + 2*SW (RW：读周期。SW：写周期)

【 指令字长 】2

【 指令版本 】全部版本

【 语 法 】CALL Label

【 例 子 】

```
.CODE
.PUBLIC    _main
_main:
r1 = 0x0001
r1 += 0x0002
call  F_Sub             //调用子程序
r1 += r4
L_Loop:
nop
jmp  L_Loop

F_Sub:
r2 = 0x5555
r3 = 0x5555
mr = r2*r3              //r2 乘 r3
retf
```

20.4.2 段间长调用指令

N	Z	S	C
-	-	-	-

【 描　　述 】长调用指令，调用地址为MR寄存器内部存的地址；PC和SR自动压入堆栈保护。

【 指令周期 】8 + RW + 2*SW

【 指令字长 】1

【 指令版本 】ISA 1.2 版本或者以上版本

【 语　　法 】CALL　MR

【 例　　子 】这个例子是用MR指向F_Sub

```
.CODE
.PUBLIC     _main
_main:
  r1 = 0xefff;
  r2 = 0x1713;
  r3 = OFFSET  F_Sub            // r3 的值为“offset”
  r4 = SEG  F_Sub               // r4 的值为 “segment”
  call  mr
L_MainLoop:
  nop
  jmp   L_ MainLoop
 F_Sub:
  r2 = 0x5555
  r3 = 0x5555
  mr = r2*r3                    //r2 乘 r3
  retf
```

20.4.3 长跳转

N	Z	S	C
-	-	-	-

【 描　　述 】跳转到用户指定的地址，这个地址不局限于64K的地址空间。

【 指令周期 】5 + 2*RW

【 指令版本 】ISA 1.1 版本或者以上版本

【 语　　法 】GOTO　LABEL;

【 例　　子 】

```
.CODE
.PUBLIC _main
```

```
_main:
        r1 = 0x000a
        r2 = 0x0001
        cmp r1,r2                  //比较 r1 和 r2 的大小

        ja  L_OK                   //r1>r2
      goto  L_Exchange_Value       //当 r1 小于等于 r2 时, 交换 r1 和 r2

 L_OK:
        nop
        jmp  L_OK

 L_Exchange_Value:
        r3 = r1
        r1 = r2                    //交换 r1 和 r2 的值
        r2 = r3
        jmp  L_OK
```

20.4.4 间接长跳转

N	Z	S	C
–	–	–	–

【 描　　述 】间接跳转到MR所指向的地址，MR中{R4[5:0],R3[15:0]} 指向用户的目的地址。

【 指令周期 】4 + RW

【 指令版本 】ISA 1.2 版本或者以上版本

【 语　　法 】GOTO　MR;

【 例　　子 】

```
.CODE
.PUBLIC  _main
_main:
   r4 =  SEG  L_GOTO_Sub          // r4 的值为 "segment"
   r3 =  OFFSET  L_GOTO_Sub       // r3 的值为"offset"
   goto  mr                       // mr(r4: r3)的值为 L_GOTO_Sub
   nop
  L_GOTO_Sub:
   nop
   jmp   L_GOTO_Sub
```

20.4.5 RETF 子程序返回

N	Z	S	C
-	-	-	-

【 描　　述 】RETF执行时会将SR和PC从堆栈中弹出并恢复，因此SR和PC会恢复到子程序调用前的状态。

【 指令周期 】8 + RW + 2*SW

【 指令字长 】1

【 指令版本 】全部版本

【 语　　法 】RETF;

【 例　　子 】

```
.CODE
.PUBLIC _main
_main:
        nop
        call  F_Delay1
L_loop:
        nop
        jmp  L_loop

F_Delay1:
        nop
        retf
```

20.4.6 RETI 中断返回

N	Z	S	C
-	-	-	-

【 描　　述 】如果在中断嵌套模式下，这条指令将弹出FR、SR、PC，否则只弹出SR和PC。从中断返回时还会清除FIQ或者是IRQ中断标志。注意，由于SR在中断返回时会出栈恢复，所以SR只在当前的中断服务中是有效的。同样FIQ中断不会打断正在运行的FIQ服务。

【 指令周期 】8 + 3*RW (IRQ 嵌套模式关闭) / 10 + 4*RW (IRQ嵌套模式打开)

【 指令字长 】1

【 指令版本 】全部版本

【 语　　法 】RETI

20.4.7 Break 软件中断

N	Z	S	C
-	-	-	-

【 描　　述 】产生软件中断，程序跳转到地址[0x00fff5]执行程序。

【 指令周期 】10 + 2*RW + 2*SW

【 指令字长 】1

【 指令版本 】全部版本

【 语　　法 】BREAK;

【 例　　子 】

```
.CODE
.PUBLIC     _main
_main:
 r1 = 0x1234
 BREAK                    //发生软件中断
L_MainLoop:
 nop
 jmp    L_ MainLoop

.TEXT
.PUBLIC      _BREAK
_BREAK:
 push  r1, r5  to  [sp]
 nop                        //在此区域里写入中断函数
 pop  r1, r5     from  [sp]
 reti                      //中断返回
```

20.4.8 FIR_MOV 开/关

N	Z	S	C
-	-	-	-

【 描　　述 】使能或禁止在FIR操作时的数据自动搬移。这会影响 FIR的操作。由于这种状态是一个全局行为，因此，用户在中断中使用它时要小心。。

【 指令周期 】2 + RW

【 指令字长 】1

【 指令版本 】全部版本

【 语　　法 】FIR_MOV ON/OFF

【 例　　子 】

```
.IRAM
```

```
.VAR NO_1=0x0001, NO_2=0x0002,   NO_3=0x0003, NO_4=0x0004
.VAR NO_5=0x0005, NO_6=0x0006,   NO_7,        NO_8
.CODE
.PUBLIC  _main
_main:
FIR_MOV ON                      //使能 FIR 操作数据自动搬移
r1 = NO_2                       //将地址“NO_2” 赋给‘r1’
r2 = NO_5                      //将地址“NO_5” 赋给‘r2
Loop_Muls:
mr = [r1]*[r2],us,2            //操作结果存入 r4: r3 中
[NO_7] =r3                     //将结果(低 16 位)读入 NO_7 中
[NO_8] = r4                    //将结果(高 16 位)读入 NO_8 中
nop
jmp   Loop_Muls
```

20.4.9 第二寄存器组选取指令

N	Z	S	C
–	–	–	–

【 描　　述 】第二寄存器组选取或是禁止指令。只有ISA1.2或是以上版本的内核才有4个映射寄存器SR1～SR4。用户可以通过 SECBANK ON/OFF 指令来选用或是禁止映射寄存器操作,在 SECBANK ON 指令执行后,所有对R1～R4的操作将全部映射到SR1～SR4。

【 指令周期 】2 + RW

【 指令字长 】1

【 指令版本 】ISA 1.2 版本或者以上版本

【 语　　法 】SECBANK ON/OFF;

【 例　　子 】

```
.CODE
.PUBLIC   _main
_main:
SECBANK  ON
r1 = 0x1234          // 0x1234 被送入 SR1 , r1 无影响
r2 = 0x5678          // 0x5678 被送入 SR2 , r2 无影响
r3 = 0x0f0f          // 0x0f0f 被送入 SR3 , r3 无影响
r4 = 0xf0f0          // 0xf0f0 被送入 SR4 , r4 无影响
L_Loop:
nop
```

```
jmp  L_Loop
```

20.4.10 Fraction 开/关模式

N	Z	S	C
-	-	-	-

【 描　　述 】Fraction开/关模式。在Fraction ON操作模式下，乘法所得到的32位结果会向左移一位。

【 指令周期 】2 + RW

【 指令字长 】1

【 指令版本 】ISA 1.2 版本或者以上版本

【 语　　法 】FRACTION ON/OFF;

【 例　　子 】

```
.CODE
.PUBLIC  _main
_main:
r1 = 0x0002
    r2 = 0x2244
      FRACTION OFF
      mr = r1*r2      //"FRACTION OFF"时，结果(R4R3)是 0x00004488

      FRACTION ON
      mr = r1*r2      //"FRACTION ON" 时，结果(R4R3)是 0x00008910
L_Loop:
  nop
  jmp   L_Loop
```

20.4.11 IRQ 嵌套模式使能或禁止指令

N	Z	S	C
-	-	-	-

【 描　　述 】 IRQ嵌套模式使能或禁止指令。 在中断嵌套模式被启用时，中断时CPU 会将FR/SR/PC 都压入堆栈保存，而在中断服务完成后将它们自动出栈恢复，如果在中断过程有更高优先级的中断发生，将发生中断嵌套。用户可以通过设置中断优先级寄存器来设置当前中断的优先级。

【 指令周期 】2 + RW

【 指令字长 】1

【 指令版本 】ISA 1.2 版本或者以上版本

【 语　　法 】IRQNEST ON/OFF;

20.4.12 IRQ 开/关

N	Z	S	C
–	–	–	–

【 描　　述 】打开或关闭 IRQ
【 指令周期 】2 + RW
【 指令字长 】1
【 指令版本 】全部版本
【 语　　法 】IRQ ON/OFF;　　//打开或关闭IRQ
【 例　　子 】

```
.include hardware.inc
.RAM
.VAR  R_Value
.CODE
.PUBLIC  _main
_main:
    r1 =  0xffff                    //将 PORTA 置为输出
    [P_IOA_Dir] = r1
    [P_IOA_Attrib] = r1
    [P_IOA_Data] = r1
    r1 =  0x0001                    //设置 2Hz 中断
    [P_INT_Ctrl] = r1
    IRQ   ON                        //打开 IRQ
L_Loop:
    nop
    jmp  L_ Loop
.TEXT
.PUBLIC  _IRQ7
_IRQ7:
push  r1,r5  to    [SP]
r4 = [R_Value]
r4 ^=    0xffff
[R_Value] =   r4
[P_IOA_Data] = r4
r1 = 0x0001                         //清除 2Hz 中断标志
[P_INT_Clear] = r1
pop   r1 ,r5  from    [SP]
reti
```

20.4.13 FIQ 开/关

N	Z	S	C
-	-	-	-

【 描　　述 】打开或关闭FIQ
【 指令周期 】 2 + RW
【 指令字长 】 1
【 指令版本 】 全部版本
【 语　　法 】FIQ ON/OFF　　// 打开或关闭 FIQ

20.4.14 中断设置

N	Z	S	C
-	-	-	-

【 描　　述 】中断设置
【 指令周期 】2 + RW
【 指令字长 】1
【 指令版本 】全部版本
【 语　　法 】

```
INT FIQ                 //使能 FIQ，禁止IRQ
INT IRQ                 //使能 IRQ，禁止FIQ
INT FIQ, IRQ            //使能 FIQ 和 IRQ
INT OFF                 //禁止FIQ 和 IRQ
```

【 例　　子 】

```
INT     FIQ             // 使能 FIQ，禁止 IRQ
INT     FIQ, IRQ        // 使能 IRQ, FIQ
```

20.4.15 DS 段地址直接访问指令

N	Z	S	C
-	-	-	-

【 描　　述 】设置或获取数据段段地址指令，因为DS段是一个 6 位寄存器，因此在进行DS段操作和零扩展时只有低 6 位数据可装入DS段中。
【 指令周期 】2 + RW
【 指令字长 】1
【 指令版本 】ISA 1.2 版本或者以上版本
【 语　　法 】

```
DS = Rs
Rs = DS
```

【 例　　子 】

```
.CODE
.PUBLIC  _main
_main:
r1 = 0xff1f                 //将 r1 的低 6 位送入 DS
DS = r1                     //可以从 “SR”寄存器中找到它
L_Loop :
nop
jmp  L_Loop
```

20.4.16　数据段段地址设置指令(6 位短立即数)

N	Z	S	C
-	-	-	-

【 描　　述 】数据段设置指令(6 位短立即数)

【 指令周期 】2 + RW

【 指令字长 】1

【 指令版本 】ISA 1.2 版本或者以上版本

【 语　　法 】DS = I6

【 例子 】

```
.CODE
.PUBLIC  _main
_main:
  DS = 0x3F          //从寄存器 “SR”中可以找到它的变化
L_Loop :
jmp  L_Loop
```

20.4.17　空操作

N	Z	S	C
-	-	-	-

【 描　　述 】空操作，PC指向下一条指令

【 指令周期 】2 + RW

【 指令字长 】1

【 指令版本 】ISA 1.2 版本或者以上版本

【 语　　法 】NOP

20.4.18 处理器标志寄存器存取指令

N	Z	S	C
–	–	–	–

【 描　　述 】直接存取处理器标志寄存器。

F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
–	AQ	BNK	FRA	FIR	SFT BUF				F	I	INE	IRQ PRI			

AQ:　　DIVS/DIVQ 除法指令执行的 AQ 标志，默认为零

BNK:　　第二寄存器组模式标志，默认为零(SECBNK ON：1，　SECBNK OFF：0)

FRA:　　FRACTION MODE，默认为零(FRACTION MODE ON：1，FRACTION MODE OFF：0)

FIR:　　FIR_MOVE MODE，默认为零(FIR_MOVE ON ：　0，FIR_MOVE OFF：1)

SFT BUF:　移位缓冲器或者 FIR 的保护位，默认为 “0000”B

F:　　FIQ 中断标志，默认为零

I:　　IRQ 中断标志，默认为零

INE:　　IRQ 嵌套模式标志，默认为零(IRQNEST ON：1，IRQNEST OFF：0)

IRQ PRI:　IRQ 优先级寄存器，复位后默认为“1000B”,如果发生任何中断，IRQ PRI 寄存器将被设为 IRQ 的优先级，只有具有更高优先级的 IRQ 才可将其中断，用户可以通过设定优先级寄存器来定制 IRQ 嵌套。

优先级顺序：IRQ0>IRQ1>IRQ2>IRQ3>IRQ4>IRQ5>IRQ6>IRQ7

IRQ 使能算法：0 - PRI -1

注：如果 FIQ 使能，FIQ 依然具有高于任何 IRQ 的优先级。

例如：

- 如果PRI为1000, 所有IRQ 0-7使能
- 如果PRI为0000, 所有IRQ 0-7非使能
- 发生 IRQ3, PRI 将被设为 0011, 只有 IRQ0-2 使能

【 指令周期 】2 + RW

【 指令字长 】1

【 指令版本 】ISA 1.2 版本或者以上版本

【 语　　法 】

```
FR = Rs ;
Rs = FR ;
```

20.4.19 除法指令 DIVS/DIVQ

N	Z	S	C
–	–	–	–

【 描　　述 】除法指令有两种DIVS 和 DIVQ，单精度除法：32位除以16位，得到16位的商，

执行时间16*3周期。除法运算可以是有符号或者是无符号的，但分子分母必需是同类型的数据。请将32位的被除数放到MR(R4，R3)中，16位的除数放到R2中，并清掉AQ标志，然后就可以进行除法运算。

对于有符号除法，首先应执行DIVS一次，从而得到商的符号位，然后多次执行DIVQ得到商；

对于无符号除法，通过多次执行DIVQ得到商。

运算结果的商在R3中。

【 指令周期 】2 + RW (DIVS) / 3 + RW (DIVQ)

【 指令字长 】1

【 指令版本 】ISA 1.2 版本或者以上版本

【 语　　法 】DIVS MR,R2;

DIVQ MR,R2;

【 例　子 1 】用 32 位有符号数(0xffff1713)除以 16 位数(divide 0x0625)

```
.CODE
.PUBLIC  _main
_main:
   r4 = 0xffff;
   r3 = 0x1713;
     r2 = 0x0625;
     r1 = 0x0001
     r4 = r4 lsl r1;          //将被除数左移 1 位形成 Q31 格式
     mr |= r3 lsl r1          //变化到下面
     r1 = fr
     clrb r1,0xe;             //清除 AQ 标志
     fr = r1;
     r1 = 0;
     DIVS mr,R2;              //除符号位
L_Signed:
     DIVQ mr,R2;              //除余数
     r1 += 1;
     cmp r1,0xf;
     jne L_Signed;
     r1 = r3;                 //商存在 R3 中,为 0xffda

L_Loop:
     nop;
     jmp  L_Loop
```

【 例子 2 】用 32 位无符号数(0x00031713)除以 16 位数 r(0x0625)

```
.CODE
.PUBLIC  _main
_main:
   r4 = 0x0003;
   r3 = 0x1713;
   r2 = 0x0625;
   r1 = 0x0001
   r4 = r4 lsl r1;          //将被除数左移 1 位形成 Q31 格式
   mr |= r3 lsl r1          //变化到下面
   r1 = fr;
   clrb r1,0xe;            // 清除 AQ 标志
   fr = r1;
   r1 = 0;
L_unsigned:
   DIVQ mr,R2;              //除余数
   r1 += 1;
   cmp r1,0x10;
   jne L_unsigned;
   r1 = r3;                //商在 R3 中, 为 0x0080

L_Loop:
   nop;
   jmp  L_Loop
```

20.4.20 EXP 指令

N	Z	S	C
–	–	–	–

【 描述 】 EXP指令用于执行规范化操作,用来统计R4寄存器中冗余的符号位的数量并将结果放入R2中。真值表如下所示：

Rsrc	Rdst	Result
SNDDDDDD	DDDDDDDD	0
SSNDDDDD	DDDDDDDD	1
SSSNDDDD	DDDDDDDD	2
SSSSNDDD	DDDDDDDD	3
SSSSSNDD	DDDDDDDD	4
SSSSSSND	DDDDDDDD	5
SSSSSSSN	DDDDDDDD	6
SSSSSSSS	NDDDDDDD	7
SSSSSSSS	SNDDDDDD	8
SSSSSSSS	SSNDDDDD	9
SSSSSSSS	SSSNDDDD	10

```
SSSSSSSS SSSSNDDD          11
SSSSSSSS SSSSSNDD          12
SSSSSSSS SSSSSSND          13
SSSSSSSS SSSSSSSN          14
SSSSSSSS SSSSSSSS          15
```

注意：

S = 符号位
N = 非符号位
D = 任何数据

【 指令周期 】2 + RW

【 指令字长 】1

【 指令版本 】ISA 1.2 版本或者以上版本

【 语　　法 】R2= EXP R4

【 例　　子 】

```
.CODE
.PUBLIC _main
_main:
    r4 = 0x1000;              // "r4" 的第 15 位是"0"
    r2 = EXP   r4             //从第 15 位开始向向前数，当遇到第一个与第 15
                               位值相反的位时，//共移动了多少位再减 1，得
                               出结果为 2
L_Loop:
    nop
    jmp   L_Loop
```

20.4.21 位操作

N	Z	S	C
–	√	–	–

【 描　　述 】位操作面向寄存器或者全部的存贮空间，源数据在操作前会影响零标志。对寄存器的位操作也一样，但如果是对寄存器进行位操作，只有低四位有效，其它位都无意义。

【 位 操 作 】位测试，位置位，位清除，位取反(test/setb/clrb/invb)

【 指令周期 】4 + RW (对寄存器的位操作) / 7 + 2*RW + SW (对任意存贮空间的位操作)

【 指令字长 】1

【 指令版本 】ISA 1.2 版本或者以上版本

【 语　　法 】

1. 对任意存贮空间的位操作

```
BITOP {DS:}[Rd], offset          //offset=0-15
BITOP {DS:}[Rd], Rs              //Rs=R0-R7     Rd=R0-R6
```

2. 对寄存器的位操作

```
BITOP Rd, offset                    //offset=0-15
BITOP Rd, Rs                        //Rs=R0-R7      Rd=R0-R6
BITOP :
```

00	01	10	11
tstb	setb	clrb	invb

【 例子 1 】

```
test  D: [r2], 13;
setb  [r1], r3;
clrb     r3, 10;
invb  r4, r5;
```

【 例子 2 】

```
.CODE
.PUBLIC  _main
_main:
 r1 = 0x0f0f
 r5 = 0x0001
 setb  r1, 15          // r1 的第 15 位置 1
 clrb  r1, 15           // r1 的第 15 位清除
 invb  r1, r5           // R5 =1, 因此对 r1 第 1 位取反 , 结果是 0x0f0d
 test  [r1], 8          // 只改变”Z”标志
L_Loop:
 nop
 jmp  L_Loop
```

20.4.22 移位操作指令

N	Z	S	C
√	√	-	-

【 描　　述 】操作只能用RD和RS进行。一个指令实现16位的多重循环移位，用两条移位指令可实现32位数据移位，此时，移位后的结果必须放在寄存器R4:R3内，右移/左移(ROR /ROL)会带着进位标志一起执行，借位标志则在移位操作后置于进位标志位上。

【 移位操作 】支持 ASR/ASROR/LSL/LSLOR/LSR/LSROR/ROL/ROR

【 指令周期 】8 + RW

【 指令字长 】1

【 指令版本 】ISA 1.2 版本或者以上版本

【 语　　法 】

Rd = Rd shift_op Rs　　Rd:　目标寄存器 (只有R0-R6可用)

Rs:　移位计数寄存器

Rs[4: 0] 合法 :　ASR, ASROR, LSL, LSLOR, LSR, LSROR

Rs[3: 0] 合法:　ROL, ROR

Shift_OP :

000	001	010	011	100	101	110	111
ASR	ASR-OR	LSL	LSL-OR	LSR	LSR-OR	ROL	ROR

【 例子 1 】

```
r2 = r2 asr r1;            // 16 位算术右移
r3 = r3 lsr r1;            // 32 位算术右移
mr |= r4 asr r1            // 结果放入 r4: r3
```

【 例子 2 】

```
r1 = 0x1234
r1 = r1 rol 4              //值为 0x2340
r1 = r1 rol 4              //值为  0x3401
r1 = r1 rol 4              //值为 0x4012
r1 = r1 rol 4              //值为 0x0123
r1 = r1 rol 4              //值为 0x1234

r1 = 0xffff
r1 = r1 lsr 4              //值为 0x0fff
r1 = r1 rol 4              /值为 0xffff
r1 = r1 lsl 4              //值为 0xfff0
r1 = r1 ror 4              //值为 0xffff
r1 |= r1 lsr 4
r1 = 0xff00              // “r1”的原始值变为 0xff00
r2 = 0x0007              // “r2” 的原始值为 0x0007
r1 = r1 asr r2           //结果为 0xfffe
```

【 例子 3 】

```
r1 = 0x0006
r3 = 0x0f0f
r4 = 0xf0f0
r3 = r3 lsr r1
mr |= r4 asr r1
```

20.4.23　寄存器乘法指令(Mul)

N	Z	S	C
-	-	-	-

【 描　　述 】二个寄存器的数据相乘，结果送到MR(R4，R3)。如果 Fraction mode 是使能的，这个结果会向左移一位。Rd只支持R0～R6。

【 指令周期 】12+RW（有符号乘有符号，无符号乘有符号）/ 13+RW（无符号乘无符号）

【 指令字长 】1

【 指令版本 】有符号乘有符号和无符号乘有符号为全部版本都有/无符号乘无符号为ISA 1.2 版本或者以上版本特有。

```
MR = Rd * Rs;          // 默认情况下为有符号乘有符号
MR = Rd * Rs,uu;       // 无符号乘无符号
MR = Rd * Rs, ss;      // 有符号乘有符号
MR = Rd * Rs, us;      // Rd 为无符号数 , Rs 为有符号数. 第一个寄存器总是无符号的
                       // 并且第二个寄存器是有符号的.
```

【 例子 1】

```
r1 = 0x8002
r2 = 0x0002            //结果存储在 R4: R3 中
mr = r1*r2             // 12 周期, r4=0xffff,r3=0x0004
mr = r1*r2,ss          // 12 周期, r4=0xffff,r3=0x0004
mr = r2*r1,us          // 12 周期, r4=0xffff,r3=0x0004
mr =r1*r2,uu           // 14 周期, r4=0x0001,r3=0x0004
```

20.4.24 乘法累加运算(Muls)

N	Z	S	C
-	-	√	-

【 描　　述 】用一个36位算术逻辑单元来进行乘-累加运算。如果 Fraction 模式使能，运行的结果会左移一位，Rd和Rs会随指令的执行而自动调整。如果FIR_MOVE 功能打开并且n>1时Rd 所指向的数据将会自动向前一地址搬移。结果的高四位保存在移位缓冲器里面，同时符号位保存着运算过程中是否有结果溢出。

【 指令周期 】10*N+6 + (N+1)*2*RW + {N*SW}　(有符号乘有符号，无符号乘有符号)

11*N+6 + (N+1)*2*RW + {N*SW}　(无符号乘无符号)

where N=1..16, (N*SW) = 0 if FIR_MOVE OFF

【 指令字长 】1

【 指令版本 】有符号乘有符号和无符号乘有符号为全部版本都有/无符号乘无符号为ISA 1.2 版本或者以上版本特有

【 语　　法 】MR = [Rd] * [Rs]{, ss}{, n}

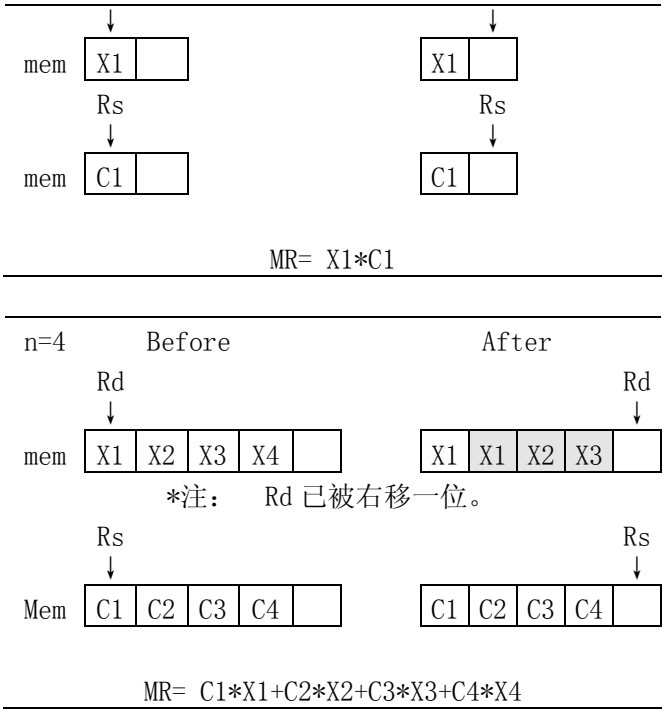
MR = [Rd] * [Rs], us{, n}

*有符号数相乘时，习惯不指出乘法属性；MR 在执行前会自动清除以前的值。

*n: [16: 1]. 如果n为1，n可以省略。

n=1　Before　After

Rd　Rd



注意： 这个 FIR 操作在 ISA1.0 和 ISA1.1 中不会改变符号标志，但在 ISA1.2 中会改变这个标志以对溢出事件做出标志。

【 **限制** 】下面的任何一种情况均会导致结果错误：

n <1 ， Rs 和 Rd 被设置为R3或者R4 ， Rs 和 Rd 设置为同一寄存器

【 **例子**】 MR = [R1] * [R2], 8

MR = [R1] * [R2], us, 2

注： 这里可以参照 FIR_MOV ON/OFF 指令的例子

20.4.25 条件分支跳转(分支)

N	Z	S	C
-	-	-	-

【 **描 述** 】短跳转指令 (当前地址±63 个字), 跳转指令和执行条件请参考表20-6

【 **指令周期** 】2+RW (not-taken) / 4+RW (taken)

【 **指令字长** 】1

【 **指令版本** 】全部版本

【 **语 法** 】

```
Conditional_jump  label
jmp label
```

【 **例 子** 】

```
jmp   LABEL;        //无条件跳转
jne   LABEL;        //不相等时跳转
```

20.4.26 压栈和出栈操作(通过间接寻址)

N	Z	S	C
–	–	–	–

【 描　　述 】将指令中标出的连续的几个寄存器压入或弹出堆栈区,同时会调整堆栈指针SP。

【 指令周期 】4+ 2*N + (N+1)*RW　　// N = 1..7

【 指令字长 】1

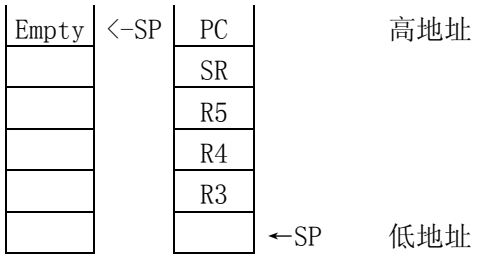
【 指令版本 】全部版本

【 语　　法 】

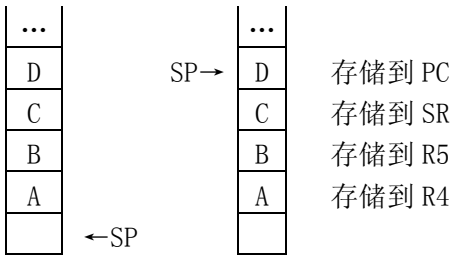
```
push Rx, Ry to [Rs]        // 将 Rx-Ry 压栈
pop Rx, Ry from [Rs] ;     // 将Rx-Ry出栈
```

【 例　　子 】

```
Push R3, PC to [SP];       // 将 R3 - PC (R7)入栈
```




```
pop R4, PC from [SP];      // 将 R4-PC (R7)出栈
pop SR, PC from [SP];      // 返回
```



注意：push R1, R5 to [SP] 等同 push R5, R1 to [SP]

20.4.27 基址变址寻址方式的 ALU 操作 (BP+6 位短立即数)

N	Z	S	C
√	√	√	√

【 描　　述 】基址变址寻址方式的ALU操作 (BP+6位短立即数), BP 就是 R5。

【 ALU　　】支持加、带进位的加、减、带借位的减、比较、逻辑与、或、异或、装载、测试、取补和存贮。

【 指令周期 】6+ 2*RW

【 指令字长 】1

【 指令版本 】全部版本

【 语　　法 】

```
Rd alu_op = [BP + IM6];
[BP + IM6] = Rd;
```

注意：列出所有可能的 ALU 操作

```
Rd -= [ BP + IM6];                  // 减法
Rd -= [ BP + IM6] , carry;          // 带借位的减法
Rd += [ BP + IM6] , carry;          // 带进位的加法
Rd += [ BP + IM6];                  // 加法
cmp Rd , [ BP + IM6];               // 比较
Rd &= [ BP + IM6];                  // 逻辑与
Rd |= [ BP + IM6];                  // 或
Rd ^= [ BP + IM6];                  // 异或
Rd = [ BP + IM6];                   // 装载
test Rd , [ BP + IM6];              // 测试
Rd = - [ BP + IM6];                 // 取补
[ BP + IM6] = Rd;                   // 存贮
```

【 例子 1】

```
r2 += [bp + 0x20], Carry;           // 相加带进位
[bp + 0x06] = r1;                   // 存储
```

【 例子 2】

```
.IRAM
T_Buffer:  .DW 0x1111, 0x1111, 0x1111, 0x1111;

……                          // bp 的初始值为 0
bp = T_Buffer;
r1    = [bp+1]              // 取得[bp+1]存储地址值 ， 并将该值赋给 r1
……
```

20.4.28　6 位立即数寻址的 ALU 操作 (Imm6)

N	Z	S	C
√	√	√	√

【 描　　述 】6 位立即数寻址的 ALU 操作

【 ALU　　】支持加、带进位的加、减、带借位的减、比较、逻辑与、或、异或、装载、测试和取补。

【 指令周期 】2+RW

【 指令字长 】1

【 指令版本 】全部版本

【 语　法 】

```
`Rd  ALU_OP =  IM6;
```

注意：列出所有可能的 ALU 操作

```
Rd -= IM6;                  //减法
Rd -= IM6 , Carry;      //带借位的减法
Rd += IM6 , Carry;      //带进位的加法
Rd += IM6;                  // 加法
cmp Rd , IM6;             //比较
Rd &= IM6;                  //逻辑与
Rd |= IM6;                  //或
Rd ^= IM6;                  //异或
Rd = IM6;                   //装载
test  Rd , IM6;             //测试
Rd =-  IM6;                 //取补
```

【 例子 1】

```
r1 += 0x20, Carry;      //r1=r1+0x20+进位
```

【 例子 2】

```
r1 = 0x00ef             // 3 cycles the origin value of r1 is 0x00ef ???
r2 = 0x0073
r3 = 0x0fff
r1 -= 0x0001            //结果变为 0x00ee
r2 += 0x0030            //结果变为 0x00a3
r3 &= 0x0002            //结果变为 0x0002
```

【例子 3】

```
r1 = 0x0002             // 3 个周期
cmp r1,0x0010           // 3 个周期比较 r1 与 0x0010
jbe L_Lower             //无符号数小于等于则跳转到 L_Lower
nop                     // 3 个周期 , 向下执行
 ……
L_Lower：
    nop
    nop
    ……
```

20.4.29　寄存器间接寻址和寄存器间接变址寻址的 ALU 操作

N	Z	S	C
√	√	√	√

【 描　　述】寄存器间接寻址和寄存器间接变址(增量可能为负)寻址的 ALU 操作。这条指令

是唯一可以寻址整个地址空间(不同的页)的指令。

【 ALU 】支持加法，带进位的加法，减法，带借位的减法，比较，逻辑与，或，异或，装载，测试，取补和存贮

【 指令周期 】6 + RW + SRW / 7 + RW + SRW (写 PC 时)

SRW：存贮或者读等待周期, 如果 ALU 执行存贮操作 SRW = SW，否则 SRW = RW

【 指令字长 】 1

【 指令版本 】 全部版本

【 语法 】

```
Rd alu_op = {D: }[Rs];
Rd alu_op = {D: }[Rs--];
Rd alu_op = {D: }[Rs++];
Rd alu_op = {D: }[++Rs];
```

【 例子 】

```
r1 &= D: [r2++];          // r1= r1 逻辑与(内容存在于 DS 和 r2 的地址里),r2=r2+1
                          // 当 r2 为 0xffff， 执行 r2= r2 + 1 操作，则 r2 变为 0 并且 DS +1
cmp  r1 , [++r2];         //令 r1 与地址为(r2+1)中的内容比较
D: [r2--] =r1;            //分配 r1 的地址为 DS 和 r2， r2=r2-1
                          // 当 r2 为 0 时, 执行 r2 - 1 操作，则 2 为 ffff 并且 DS -1
```

20.4.30 16 位立即数寻址的 ALU 操作

N	Z	S	C
√	√	√	√

【 描 述 】16 位立即数寻址的 ALU 操作

【 ALU 】支持加法，带进位的加、减、带借位的减、比较、逻辑与、或、异或、装载、测试和取补。

【 指令周期 】4+2*RW / 5+2*RW (write to PC)

【 指令字长 】2

【 指令版本 】全部版本

【 语 法 】

```
Rd = Rs alu_op IM16;
```

【 例子 2】

```
.CODE
    .PUBLIC  _main
    _main:
```

```
                r1 = 0xffff
                r2 = 0
L_Imm16:
                cmp r1, 0x00ff              //将 r1 与 16 位立即数 0x00ff 比较
                jb  L_Loop
                r1 -= 0x00ff               //r1 与 0x00ff 相减
                r2 += 1
                goto L_Imm16
L_Loop:
                jmp     L_Loop
```

20.4.31 16 位直接地址寻址的 ALU 操作

N	Z	S	C
√	√	√	√

【 描　　述 】16位直接地址寻址的ALU操作

【　ALU　】支持加、带进位的加、减、带借位的减、比较、逻辑与、或、异或、装载、测试和取补。

【 指令周期 】7 + 2*RW + SRW / 8 + 2*RW + SRW (write to PC)

【 指令字长 】2

【 指令版本 】全部版本

【 语　　法 】

```
Rd = Rs alu_op [A16]
[A16] = Rs alu_op Rd
```

【 例子1】

```
bp = r1 + [Q_table];          // bp =r1 + Q_table的内容 , Q _table位于 0x0000 ～
                                 0xffff 之间
[Hashing_tbl] = -bp;          // 将 - bp 的值赋予 Hashing_tbl 单元
[Q_Table+10] = bp ^ r2;       // bp 异或 r2 的结果存入 Q_Table+10 单元
r1 = [0x8079];                //取得地址为 0x8079 存储单元的值 , 将其存入 r1
```

20.4.32 6 位直接地址寻址的 ALU 操作

N	Z	S	C
√	√	√	√

【 描　　述 】6位直接地址寻址的ALU操作

【　ALU　】支持加、带进位的加、减、带借位的减、比较、逻辑与、或、异或、装载、测试和取补。

【 指令周期 】5+ RW +SRW / 6+ RW +SRW (写PC寄存器)

【 指令字长 】1

【 指令版本 】全部版本

【 语　　法 】

```
Rd = Rd alu_op [A6]
Rd = [A6]
[A6] = Rd
```

【例子 1】

```
.RAM
.VAR  Init_Q ;                    // Init_Q 是一个范围在 0x00～0x3f 可变
的地址值
.CODE
.public _main
_main:
      r1=  [0x0009]               //取得地址为 0x0009 的单元的内容, 并将
其送入 r1
   r2 = 0x5555;
   bp = [Init_Q];                 //将 Init_Q 单元的内容载入 bp
   [Init_Q] = R2;                 //将 r2 的值存储到 Init_Q 单元中去
    ……
```

20.4.33　寄存器寻址的 ALU 操作

N	Z	S	C
√	√	√	√

【 描　　述 】寄存器寻址的ALU操作

【　ALU　 】支持加、带进位的加、减、带借位的减、比较、逻辑与、或、异或、装载、测试、取补和存贮。

【 指令周期 】3 + RW / 5 + RW (write to PC)

【 指令字长 】1

【 指令版本 】全部版本

【 语　　法 】

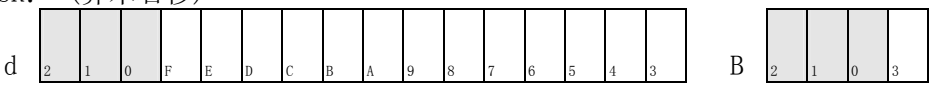
```
Rd {alu_op} =  Rs {, Carry}
Rd {alu_op} =  Rs ASR n {, Carry}
Rd {alu_op} =  Rs LSL n {, Carry}
Rd {alu_op} =  Rs LSR n {, Carry}
Rd {alu_op} =  Rs ROL n {, Carry}
Rd {alu_op} =  Rs ROR n {, Carry}
```

n 为移位的次数，其范围为(1--4)

Input:　原始数据：

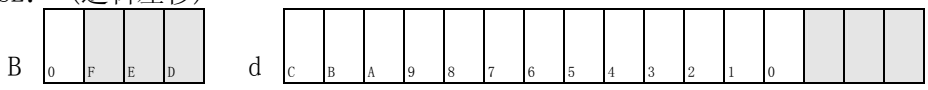
假设 n=3，移位操作后的结果：

ASR：（算术右移）

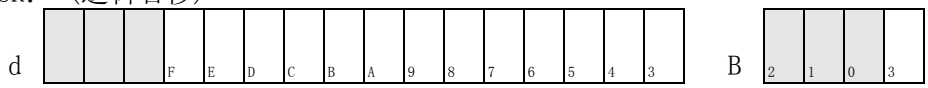


$E_2E_1E_0$ 是 Rs 的符号扩展位

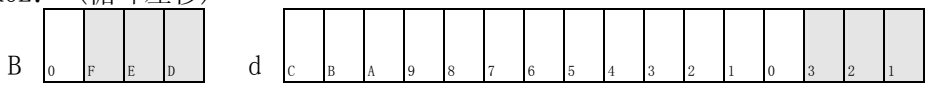
LSL：（逻辑左移）



LSR：（逻辑右移）



ROL：（循环左移）



ROR：（循环右移）

注意 ：

1. FIQ, IRQ 有自己特有的移位缓冲寄存器，在中断发生时用户不需要保存它。
2. 在乘法累加运算和 FIR 运算执行后，移位缓冲器的值是被改变的。

【 例子 】

```
bp += r1 asr 4, Carry;          // bp = bp + (r1/(2^4)) 带进位
bp = r1 lsl 2;                  // bp = r1 左移 2 位
```

第21章 硬件概述

21.1 系统控制寄存器

P_Reset_Status(R/W) ($7006)：此寄存器用于软件检测复位状态的标志

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
0	0	0	0	0	0	0	0
FCHK							-

B7	B6	B5	B4	B3	B2	B1	B0
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
保留	IIRF	IARF	SPLVRF	LPLVRF	WDRF	PORF	EXTRF

P_Clk_Ctrl ($7007)：系统时钟控制寄存器

B15	B14	B13	B2	B11	B10	b9	B8
R/W	R/W	R	R	R	R	R	R
0	0	0	0	0	0	0	0
OSCSF	OSCIE	保留					

B7	B6	B5	B4	B3	Bb2	b1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

21.2 看门狗控制寄存器

P_WatchDog_Ctrl (R/W) ($700A)：看门狗控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R	R	R	R	R	R
0	0	0	0	0	0	0	0
WDEN	WDRS	保留					

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
WDCHK					WDPS		

P_WatchDog_Clr (W) ($700B)：看门狗清除寄存器

B15	B14	B13	B12	B11	B10	B9	B8
W	W	W	W	W	W	W	W
0	1	0	1	1	0	1	0
WDTCLR							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	W
0	1	0	1	1	0	1	0
WDTCLR							

21.3 节电模式控制寄存器

P_Wait_Enter ($700C)：Wait 模式入口寄存器

B15	B14	B13	B12	B11	B10	B9	B8
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
WaitCMD							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	R/W
0	0	0	0	0	0	0	0
WaitCMD							

P_Stdby_Enter ($700E)： Standby 模式入口寄存器

B15	B14	B13	B12	B11	B10	B9	B8
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
StdbyCMD							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	R/W
0	0	0	0	0	0	0	0
StdbyCMD							

21.4 I/O 端口的控制寄存器

P_IOA_Data ($7060H)：端口 A 的数据寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Data							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Data							

P_IOA_Buffer ($7061H)：端口 A 的缓冲寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Buffer							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Buffer							

P_IOA_Dir ($7062H)：端口 A 的方向寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Dir							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Dir							

P_IOA_Attrib ($7063H)：端口 A 的属性寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

0	0	0	0	0	0	0	0
P_IOA_Attrib							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Attrib							

P_IOA_Latch ($7064H)：端口 A 的锁存寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
P_IOA_Latch							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

P_IOA_SPE ($7080H)：端口 A 的特殊功能使能寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R/W	R/W	R/W	R/W	R/W	R/W	R
0	0	0	0	0	0	0	0
保留	TCLKDEN	TCLKCEN	TCLKBEN	TCLKAEN	TIO2BEN	TIO2AEN	保留

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

P_IOA_KCER ($7084H)： IO 端口 A 的按键唤醒使能寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
KC15EN	KC14EN	KC13EN	KC12EN	KC11EN	KC10EN	KC9EN	KC8WE

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R

0	0	0	0	0	0	0	0
保留							

P_IOB_Data ($7068H)： IO 端口 B 的数据寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Data							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Data							

P_IOB_Buffer ($7069H)： IO 端口 B 的缓冲寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Buffer							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Buffer							

P_IOB_Dir ($706AH)： IO 端口 B 的方向寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Dir							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Dir							

P_IOB_Attrib ($706BH)：IO 端口 B 的属性寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Attrib							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Attrib							

P_IOB_SPE ($7081H)： IO 端口 B 的特殊功能使能寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
保留					TIO0AEN	TIO0BEN	TIO0CEN

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	1	1	1	1	1	1
OL1EN	FTIN1EN	U1EN	V1EN	W1EN	U1NEN	V1NEN	W1NEN

P_IOC_Data ($7070H)： IO 端口 C 的数据寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Data							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Data							

P_IOC_Buffer ($7071H)： IO 端口 C 的缓冲寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

P_IOC_Buffer

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Buffer							

P_IOC_Dir ($7072H)： IO 端口 C 的方向寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Dir							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Dir							

P_IOC_Attrib ($7073H)： IO 端口 C 的属性寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Attrib							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Attrib							

P_IOC_SPE ($7082H)： IO 端口 C 的特殊功能使能寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	0	0
W2NEN	V2NEN	U2NEN	W2EN	V2EN	U2EN	FTIN2EN	OL2EN

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TIO1CEN	TIO1BEN	TIO1AEN	保留	EXINT1EN	EXINT0EN	保留	

P_IOD_Data ($7078H)： IO 端口 D 的数据寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Data							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Data							

P_IOD_Buffer ($7079H)： IO 端口 D 的缓冲寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Buffer							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Buffer							

P_IOD_Dir ($707AH)： IO 端口 D 的方向寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Dir							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Dir							

P_IOD_Attrib ($707BH)： IO 端口 D 的属性寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

0	0	0	0	0	0	0	0
P_IOD_Attrib							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Attrib							

21.5 中断模块控制寄存器

P_INT_Status ($70A0)： 中断状态寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
KEYIF	UARTIF	SPIIF	EXT1IF	EXT0IF	ADCIF	MCP4IF	MCP3IF

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
TPM2IF	PDC1IF	PDC0IF	CMTIF	保留	OLIF	OSCSF	FTIF

P_INT_Priority ($70A4)： IRQ 与 FIQ 的优先权选择寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
KEYIP	UARTIP	SPIIP	保留	EXTIP	ADCIP	MCP4IP	MCP3IP

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R	R	R/W	R/W
0	0	0	0	0	0	0	0
TPM2IP	PDC1IP	PDC0IP	CMTIP	保留	OLIP	OSCIP	FTIP

P_MisINT_Ctrl ($70A8)：综合中断控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R	R	R
0	0	0	0	0	0	0	0
KEYIE	EXT1MS	EXT0MS	EXT1IE	EXT0IE	保留		

B7	B6	B5	B4	B3	B2	B1	B0

R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

21.6 通用定时器模块控制寄存器

P_TMR0_Ctrl ($7400)：PDC 定时器 0 的控制寄存器

P_TMR1_Ctrl ($7401)：PDC 定时器 1 的控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SPCK		MODE				CLEGS	

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CCLS			CKEGS		TMRPS		

P_TMR2_Ctrl ($7402)：TPM 定时器 2 的控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SPCK		MODE				CLEGS	

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CCLS			CKEGS		TMRPS		

P_TMR3_Ctrl ($7403)：MCP 定时器 3 的控制寄存器

P_TMR4_Ctrl ($7404)：MCP 定时器 4 的控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
PRDINT		MODE				保留	

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

CCLS	CKEGS	TMRPS

P_TMR0_TCNT ($7430)：PDC 定时器 0 的计数寄存器
P_TMR1_TCNT ($7431)：PDC 定时器 1 的计数寄存器
P_TMR2_TCNT ($7432)：TPM 定时器 2 的计数寄存器
P_TMR3_TCNT ($7433)：MCP 定时器 3 的计数寄存器
P_TMR4_TCNT ($7434)：MCP 定时器 4 的计数寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
TMRCNT							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
TMRCNT							

P_TMR0_TGRA ($7440)：PDC 定时器 0 的通用寄存器 A
P_TMR0_TGRB ($7441)：PDC 定时器 0 的通用寄存器 B
P_TMR0_TGRC ($7442)：PDC 定时器 0 的通用寄存器 C
P_TMR1_TGRA ($7440)：PDC 定时器 1 的通用寄存器 A
P_TMR1_TGRB ($7441)：PDC 定时器 1 的通用寄存器 B
P_TMR1_TGRC ($7442)：PDC 定时器 1 的通用寄存器 C
P_TMR2_TGRA ($7440)：TPM 定时器 2 的通用寄存器 A
P_TMR2_TGRB ($7441)：TPM 定时器 2 的通用寄存器 B
P_TMR3_TGRA ($7440)：MCP 定时器 3 的通用寄存器 A
P_TMR3_TGRB ($7441)：MCP 定时器 3 的通用寄存器 B
P_TMR3_TGRC ($7442)：MCP 定时器 3 的通用寄存器 C
P_TMR4_TGRA ($7440)：MCP 定时器 4 的通用寄存器 A
P_TMR4_TGRB ($7441)：MCP 定时器 4 的通用寄存器 B
P_TMR4_TGRC ($7442)：MCP 定时器 4 的通用寄存器 C

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRGLR							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

0	0	0	0	0	0	0	0
TMRGLR							

P_TMR0_TPR ($7435)：PDC 定时器 0 的周期寄存器
P_TMR1_TPR ($7436)：PDC 定时器 1 的周期寄存器
P_TMR2_TPR ($7437)：TPM 定时器 2 的周期寄存器
P_TMR3_TPR ($7438)：MCP 定时器 3 的周期寄存器
P_TMR4_TPR ($7439)：MCP 定时器 4 的周期寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
TMRPRD							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
TMRPRD							

P_TMR0_TBRA ($7450)：PDC 定时器 0 的缓冲寄存器 A
P_TMR0_TBRB ($7451)：PDC 定时器 0 的缓冲寄存器 B
P_TMR0_TBRC ($7452)：PDC 定时器 0 的缓冲寄存器 C
P_TMR1_TBRA ($7453)：PDC 定时器 1 的缓冲寄存器 A
P_TMR1_TBRB ($7454)：PDC 定时器 1 的缓冲寄存器 B
P_TMR1_TBRC ($7455)：PDC 定时器 1 的缓冲寄存器 C
P_TMR2_TBRA ($7456)：TPM 定时器 2 的缓冲寄存器 A
P_TMR2_TBRB ($7457)：TPM 定时器 2 的缓冲寄存器 B
P_TMR3_TBRA ($7458)：MCP 定时器 3 的缓冲寄存器 A
P_TMR3_TBRB ($7459)：MCP 定时器 3 的缓冲寄存器 B
P_TMR3_TBRC ($745A)：MCP 定时器 3 的缓冲寄存器 C
P_TMR4_TBRA ($745C)：MCP 定时器 4 的缓冲寄存器 A
P_TMR4_TBRB ($745D)：MCP 定时器 4 的缓冲寄存器 B
P_TMR4_TBRC ($745E)：MCP 定时器 4 的缓冲寄存器 C

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRBUF							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRBUF							

P_TMR0_IOCtrl ($7410) ： PDC 定时器 0 输入输出控制寄存器

P_TMR1_IOCtrl ($7411) ： PDC 定时器 1 输入输出控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
保留				IOCMOD			

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
IOBMOD				IOAMOD			

P_TMR2_IOCtrl ($7412) ： TPM 定时器 2 输入输出控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
IOBMOD				IOAMOD			

P_TMR3_IOCtrl ($7413) ： MCP 定时器 3 输出控制寄存器

P_TMR4_IOCtrl ($7414) ： MCP 定时器 4 输出控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
保留				IOCMOD			

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

IOBMOD	IOAMOD

P_TMR0_INT ($7420)：PDC 定时器 0 的中断使能寄存器

P_TMR1_INT ($7421)：PDC 定时器 1 的中断使能寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R/W
0	0	0	0	0	0	0	0
保留							PDCIE

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TADSE	TCUIE	TCVIE	TPRIE	保留	TGCIE	TGBIE	TGAIE

P_TMR2_INT ($7422)： TPM 定时器 2 中断使能寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R	R	R/W	R	R	R/W	R/W
0	0	0	0	0	0	0	0
TADSE	保留		TPRIE	保留		TGBIE	TGAIE

P_TMR3_INT ($7423)：MCP 定时器 3 的中断使能寄存器

P_TMR4_INT ($7424)：MCP 定时器 4 的中断使能寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R	R	R/W	R/W	R	R	R
0	0	0	0	0	0	0	0
TADSE	保留		TPRIE	TGDIE	保留		

P_TMR0_Status ($7425)：PDC 定时器 0 的中断状态寄存器

P_TMR1_Status ($7426)：PDC 定时器 1 的中断状态寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R/W
0	0	0	0	0	0	0	0
保留							PDCIF

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TCDF	TCUIF	TCVIF	TPRIF	保留	TGCIF	TGBIF	TGAIF

P_TMR2_Status ($7427)：TPM 定时器 2 的中断状态寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R	R	R/W	R	R	R/W	R/W
0	0	0	0	0	0	0	0
TCDF	保留		TPRIF	保留		TGBIF	TGAIF

P_TMR3_Status ($7428)：MCP 定时器 3 的中断状态寄存器

P_TMR4_Status ($7429)：MCP 定时器 4 的中断状态寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R	R	R/W	R/W	R	R	R
0	0	0	0	0	0	0	0
TCDF	保留		TPRIF	TGDIF	保留		

P_TMR_Start ($7405)：定时器计数开始寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R

0	0	0	0	0	0	0	0
保留							
B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
保留			TMR4ST	TMR3ST	TMR2ST	TMR1ST	TMR0ST

P_TMR_Output ($7406)：定时器输出使能寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
保留		TMR4FOE	TMR4EOE	TMR4DOE	TMR4COE	TMR4BOE	TMR4AOE

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
保留		TMR3FOE	TMR3EOE	TMR3DOE	TMR3COE	TMR3BOE	TMR3AOE

P_TMR_LDOK ($740A) ： MCP 定时器同步载入寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
TLDCHK						LDOK1	LDOK0

P_TMR3_OutputCtrl ($7407)：MCP 定时器 3 的输出控制寄存器

P_TMR4_OutputCtrl ($7408)：MCP 定时器 4 的输出控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	RW	R	R	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
DUTYMODE	POLP	保留			WPWM	VPWM	UPWM

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

SYNC	WOC	VOC	UOC

P_TPWM_Write ($7409)：MCP 定时器/PWM 模块写入使能控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R/W	R/W
0	0	0	0	0	0	0	0
保留						TMR4WE	TMR3WE

P_POS0_DectCtrl ($7462)：PDC 定时器 0 位置侦测数据寄存器

P_POS1_DectCtrl ($7463)：PDC 定时器 1 位置侦测数据寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	RW	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SPLCK		SPLMOD		SPLCNT			

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
PDEN	SPDLY						

P_POS0_DectData ($7464)：PDC 定时器 0 的位置侦测数据寄存器

P_POS1_DectData ($7465)：PDC 定时器 1 的位置侦测数据寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留					PDR		

P_TMR3_DeadTime ($7460)：MCP 定时器 3 的死区时间与控制寄存器

P_TMR4_DeadTime ($7461)：MCP 定时器 4 的死区时间与控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R/W	RW	R/W	R	R	R	R
0	0	0	0	0	0	0	0
保留	DTWE	DTVE	DTUE	保留			

B7	B6	B5	B4	B3	B2	B1	B0
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
保留	DTP						

P_ Faul1_Ctrl ($7466)：故障输入 1 的控制与状态寄存器

P_Fault2_Ctrl ($7467)：故障输入 2 的控制与状态寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	RW	R/W	R/W	R	R	R	R
0	0	0	0	0	0	0	0
OCE	OCIE	OCLS	OSF	保留			

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
FTPINE	FTPINIE	FTPINIF	保留	FTCNT			

P_OL1_Ctrl($7468)：过载输入 1 的控制与状态寄存器

P_OL2_Ctrl($7469)：过载输入 2 的控制与状态寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	RW	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
OLEN	CNTSP	OLMD		OLST	RTTMB	RTPWM	RTOL

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
OLIE	OLIF	保留		OLCNT			

P_Fault1_Release($746A)：故障 1 的标志释放寄存器

P_Fault2_Release($746B)：故障 2 的标志释放寄存器

B15	B14	B13	B12	B11	B10	B9	B8
W	W	W	W	W	W	W	W

0	0	0	0	0	0	0	0
FTRR							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
FTRR							

21.7 CMT 定时器控制寄存器

P_CMT_Start($7500)：比较匹配定时器开始寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R/W	R/W
0	0	0	0	0	0	0	0
保留						ST1	ST0

P_CMT_Ctrl($7500)： 比较匹配定时器控制与状态寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R	R	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CM1IF	CM1IE	保留			CKB		

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R	R	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CM0IF	CM0IE	保留			CKA		

P_CMT0_TCNT($7508)：比较匹配定时器计数器 0 的寄存器

P_CMT1_TCNT($7509)：比较匹配定时器计数器 1 的寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0

CMTCNT

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
CMTCNT							

P_CMT0_TPR($7510)： 比较匹配定时器 0 的周期寄存器

P_CMT1_TPR($7511)： 比较匹配定时器 1 的周期寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CMTPR							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CMTPR							

21.8 TMB/蜂鸣器控制寄存器

P_TMB_Reset($70B8)：时基复位寄存器

B15	B14	B13	B12	B11	B10	B9	B8
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
TBRR							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
TBRR							

P_BZO_Ctrl($70B9)：蜂鸣器输出控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
BZOEN	保留						

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R/W	R/W
0	0	0	0	0	0	0	0
保留							BZOCK

21.9 SPI 模块控制寄存器

P_SPI_Ctrl($7140)： SPI 控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R	R	R	W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SPIE	保留			SPIRST	SPISPCLK		SPIMS

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
保留		SPIPHA	SPIPOL	SPISMPS	SPIFS		

P_SPI_TxStatus($7141)： SPI 发送状态寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R	R	R	R	R
0	0	0	0	0	0	0	0
SPITXIF	SPITXIE	SPITXBF	保留				

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

P_SPI_TxBuf($7142)： SPI 传输缓冲寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

SPITXBUF

P_SPI_RxStatus($7143)： SPI 接收状态寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R	R	R	R/W	R	R
0	0	0	0	0	0	0	0
SPIRXIF	SPIRXIE	保留			FERR	保留	
B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

P_SPI_RxBuf($7144)： SPI 接收缓冲寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SPIRXBUF							

21.10 UART 模块控制寄存器

P_UART_Data($7100)： UART 数据寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留				OE	保留	FE	PE

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
UARTDATA							

P_UART_RXStatus($7101)： UART 接收状态寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	Bb2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
保留				OE	保留	FE	PE

P_UART_Ctrl($7102)：UART 控制寄存器

B15	B14	B3	B12	B11	B10	B9	B8
R/W	R/W	R	R/W	W	R/W	R/W	R
0	0	0	0	0	0	0	0
RXIE	TXIE	保留	UEN	Reset	TXCHSEL	RXCHSEL	保留

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R/W	R/W	R/W	R
0	0	0	0	0	0	0	0
保留				SBSEL	PSEL	PEN	保留

P_UART_BaudRate($7103)： UART 波特率设定寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
UARTBUD							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
UARTBUD							

P_UART_Status($7104)： UART 状态寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
RXIF	TXIF	保留					

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R

0	0	0	0	0	0	0	0
保留	RXBF	保留		BY	保留		

21.11 ADC 模块控制寄存器

P_ADC_Setup($7160)： ADC 设定寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
1	0	0	0	0	0	0	0
ADCCS	ADCEN	保留			ADCFS		ADCEXTRG

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
ASPEN	保留						

P_ADC_Ctrl($7161)： ADC 控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R	R	R	R	R	R
0	0	0	0	0	0	0	0
ADCIF	ADCIE	保留					

B7	B6	B5	B4	B3	B2	B1	B0
R	R/W	R	R	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
ADCRDY	ADCSTR	保留			ADCCHS		

P_ADC_Channel($7166)： ADC 输入通道使能寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
保留							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
ADCCH7	ADCCH6	ADCCH5	ADCCH4	ADCCH3	ADCCH2	ADCCH1	ADCCH0

P_ADC_Data($7166)： ADC 数据寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
ADCDATA							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
ADCDATA		保留					

21.12 Flash 模块控制寄存器

P_Flash_RW ($704D)：Flash 访问控制寄存器

B15	B14	B13	B12	B11	B10	B9	B8
R	R/W	R	R	R	R	R	R
1	0	1	1	1	1	1	1
保留	BK14WENB	保留					

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
1	1	1	1	1	1	1	1
保留							

P_Flash_CMD ($7555)：内嵌的 Flash 控制命令寄存器

B15	B14	B13	B12	B11	B10	B9	B8
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
FlashCtrl							

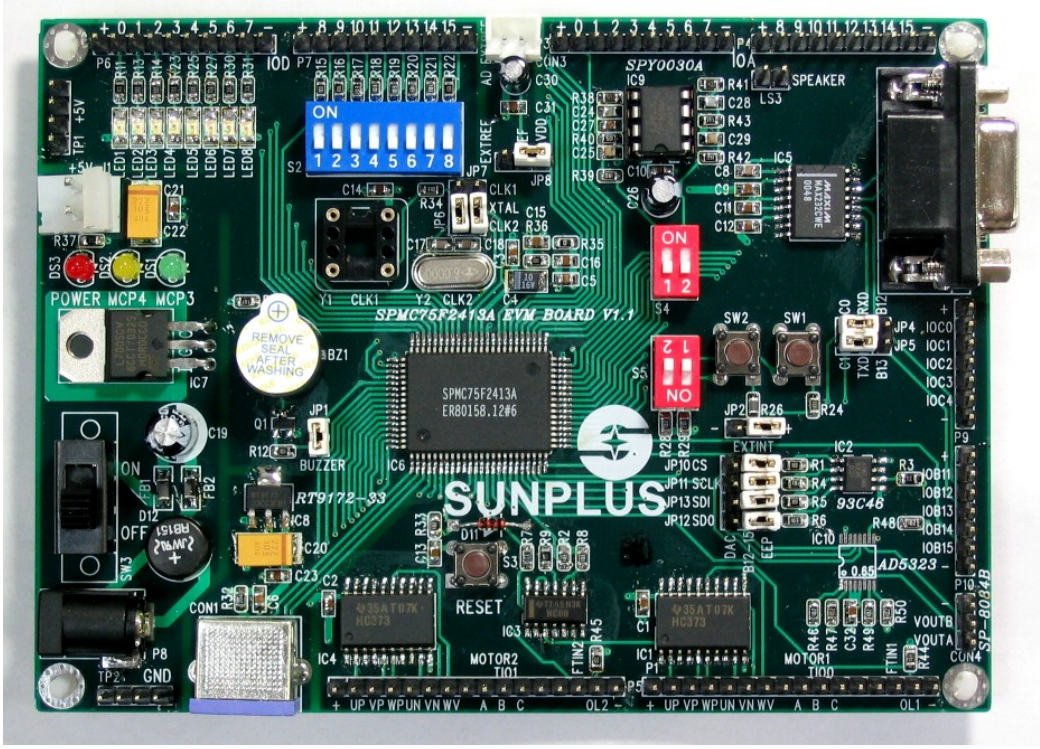
B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
FlashCtrl							

第22章 SPMC75X 系列微控制器开发系统

22.1 开发系统设置示意图

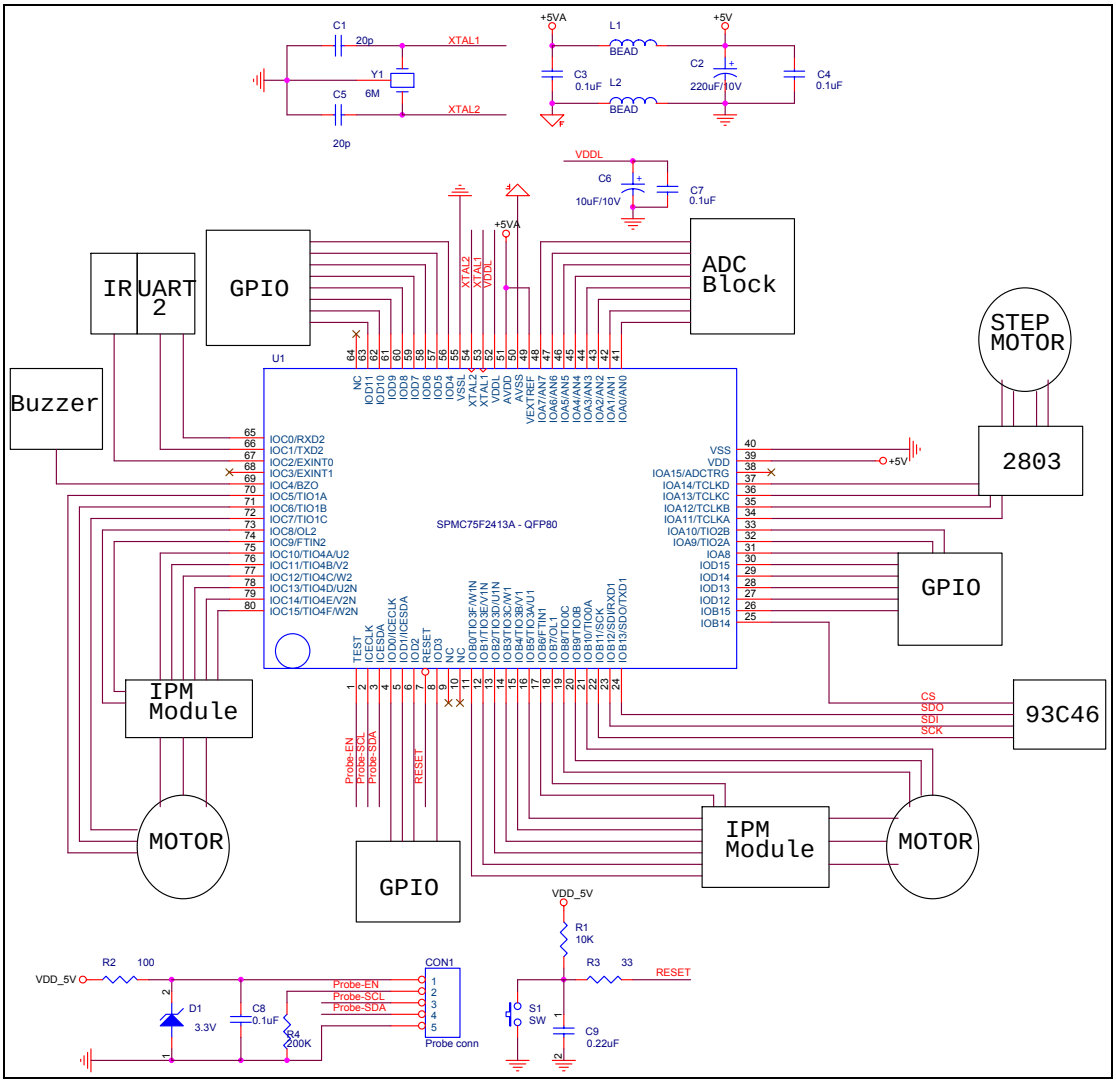
连接：PC 并口/USB 端口→ SunplusIT Probe → SPMC75F2413A EVM 板

22.2 SPMC75X 系列微控制器 EVM 板 v1.1

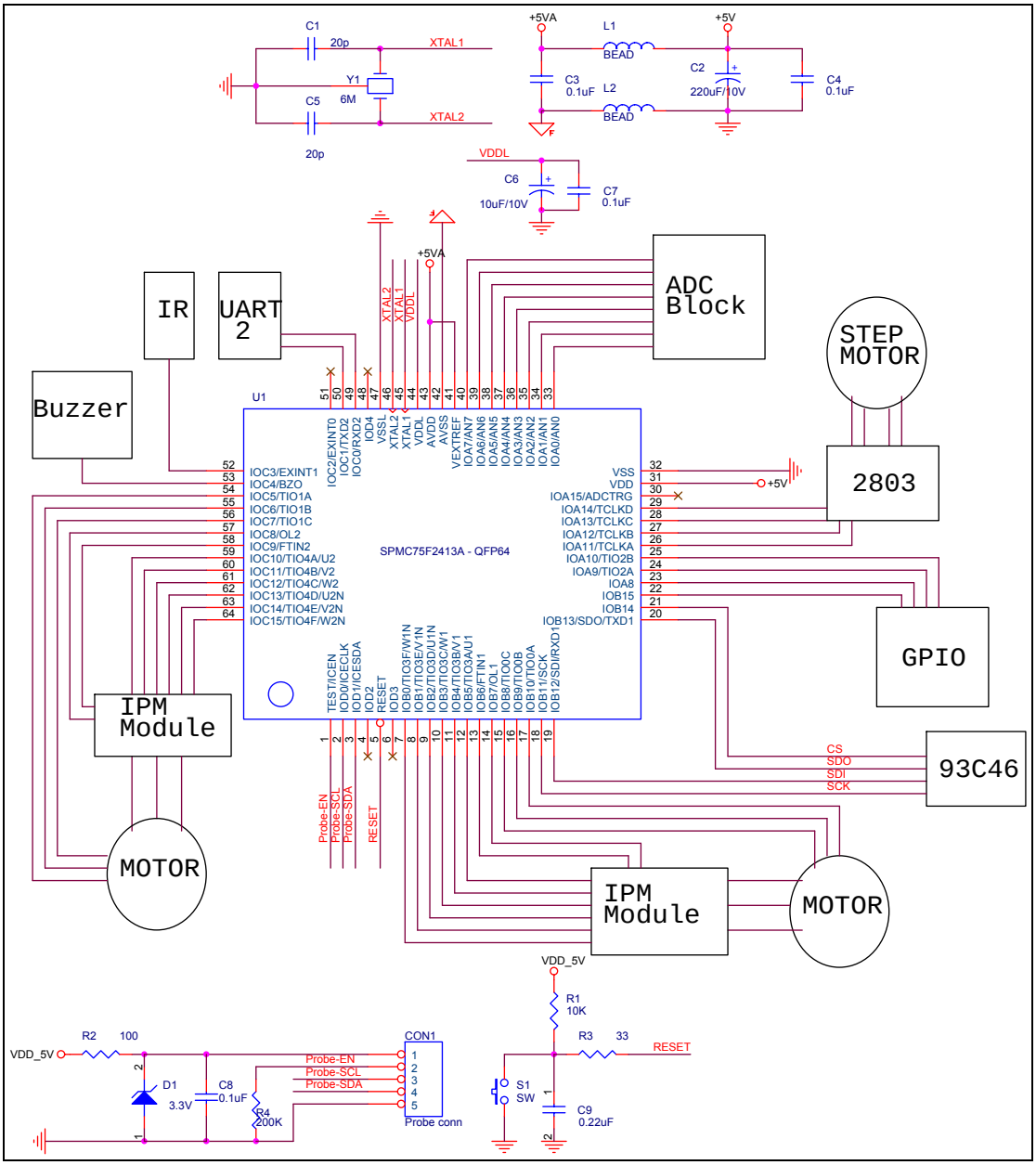


22.3 SPMC75X 系列微控制器的应用电路

22.3.1 SPMC75F2413A QFP80 引脚应用



22.3.2 SPMC75F2413A QFP64 引脚应用



22.4 SPMC75F2413A EVM 板 v1.1 示意图

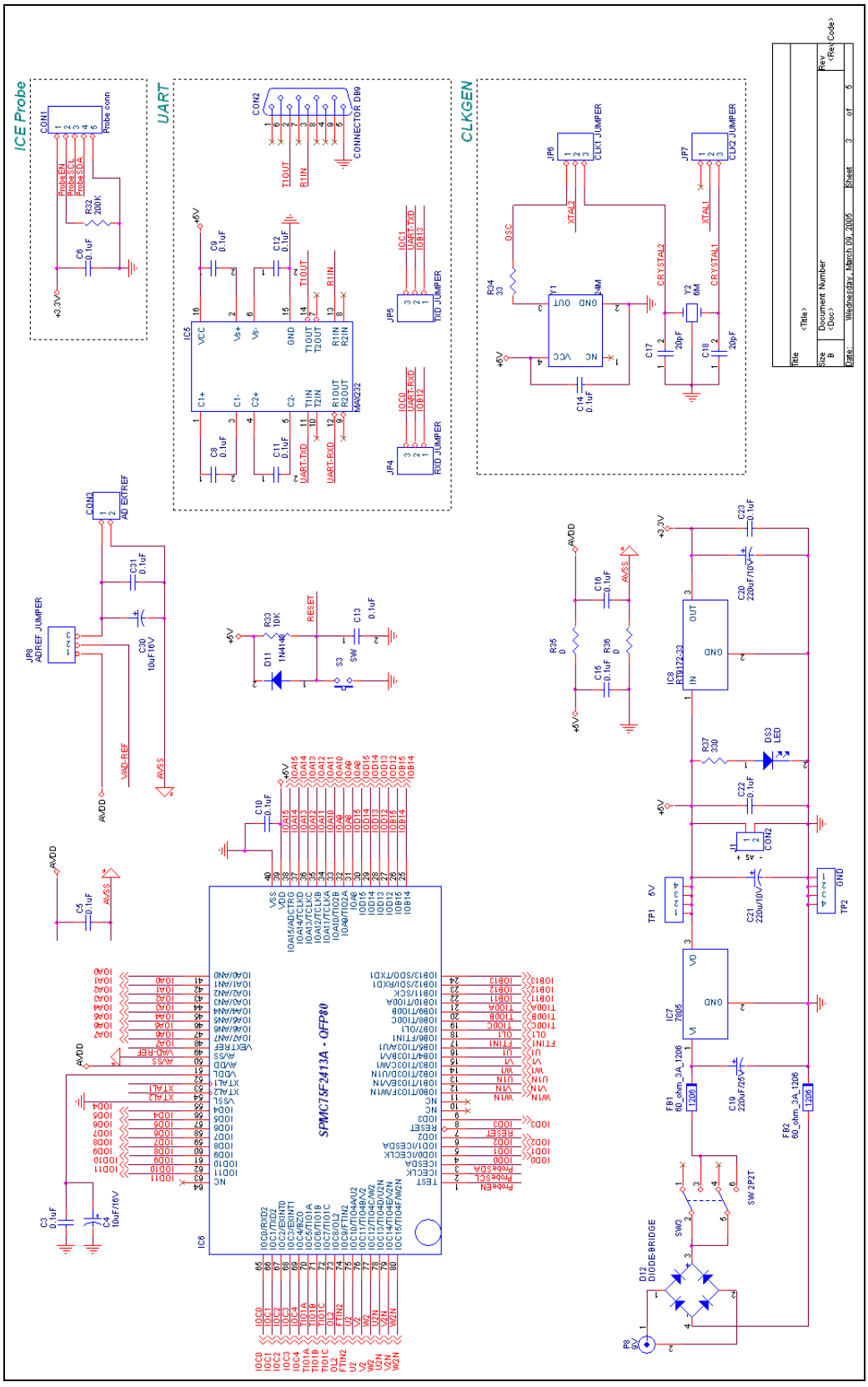


图22-1 SPMC75F2413A EVM 板原理-1

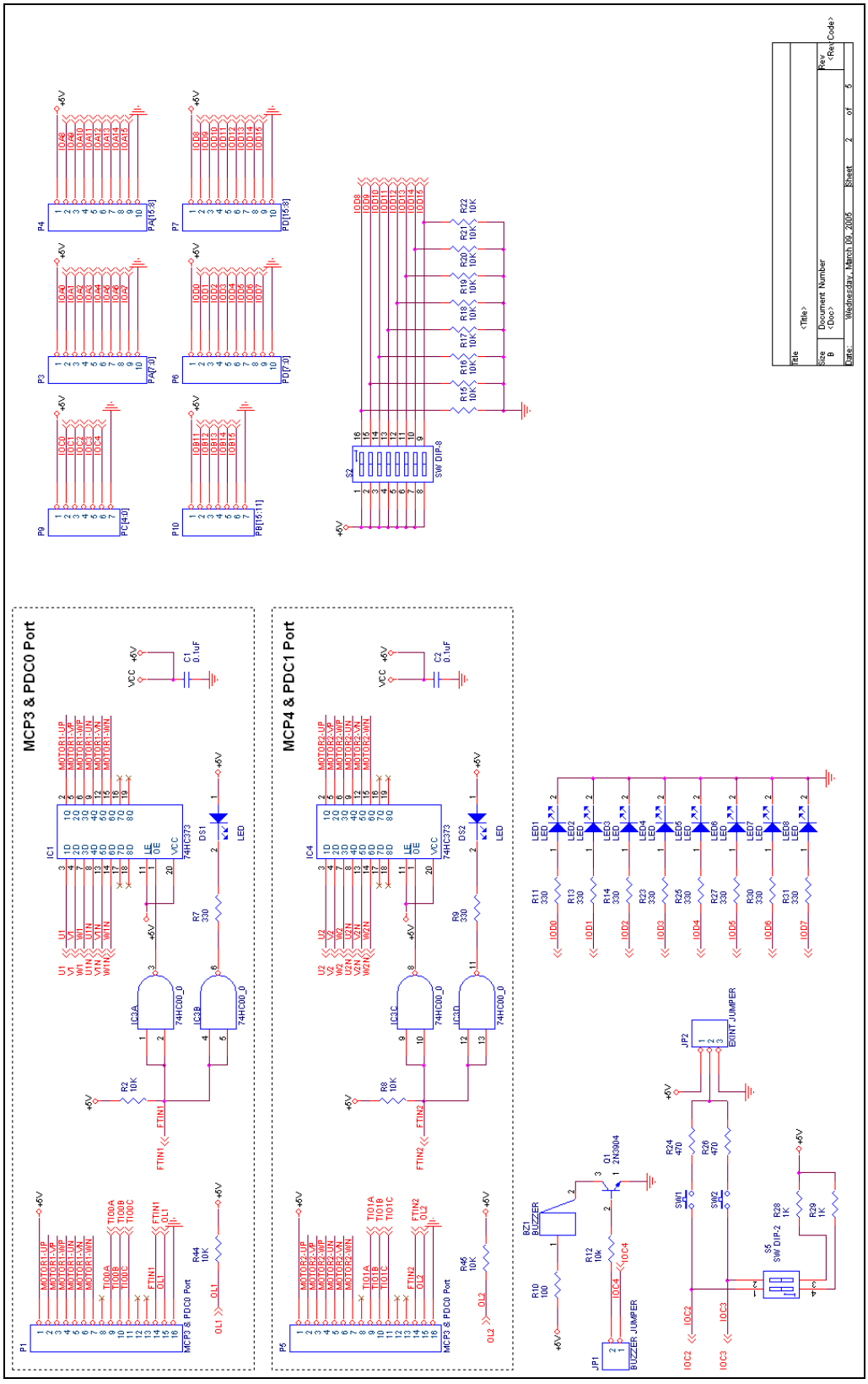


图22-2　SPMC75F2413A EVM 板原理-2

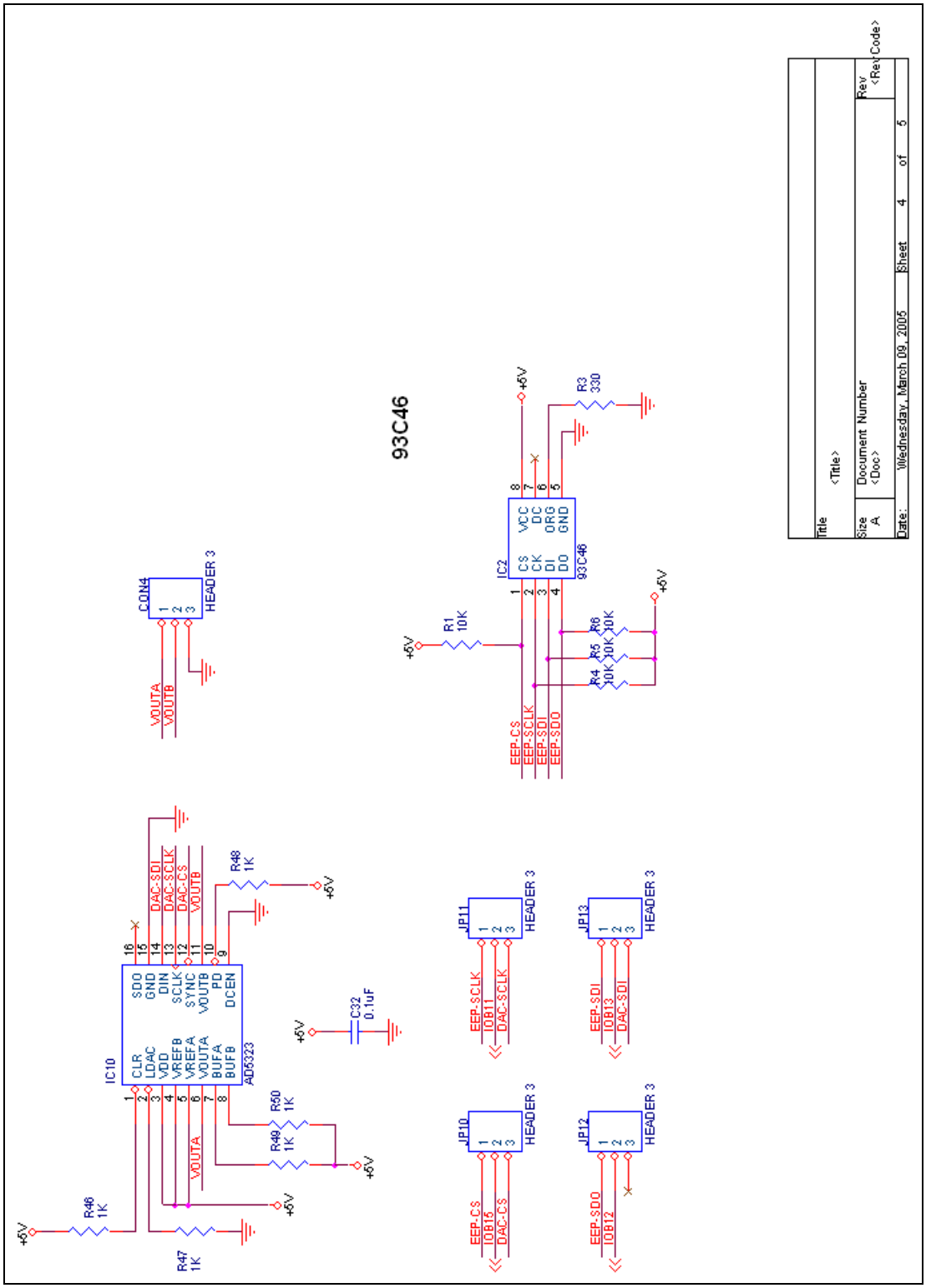


图22-3 SPMC75F2413A EVM 板原理-3

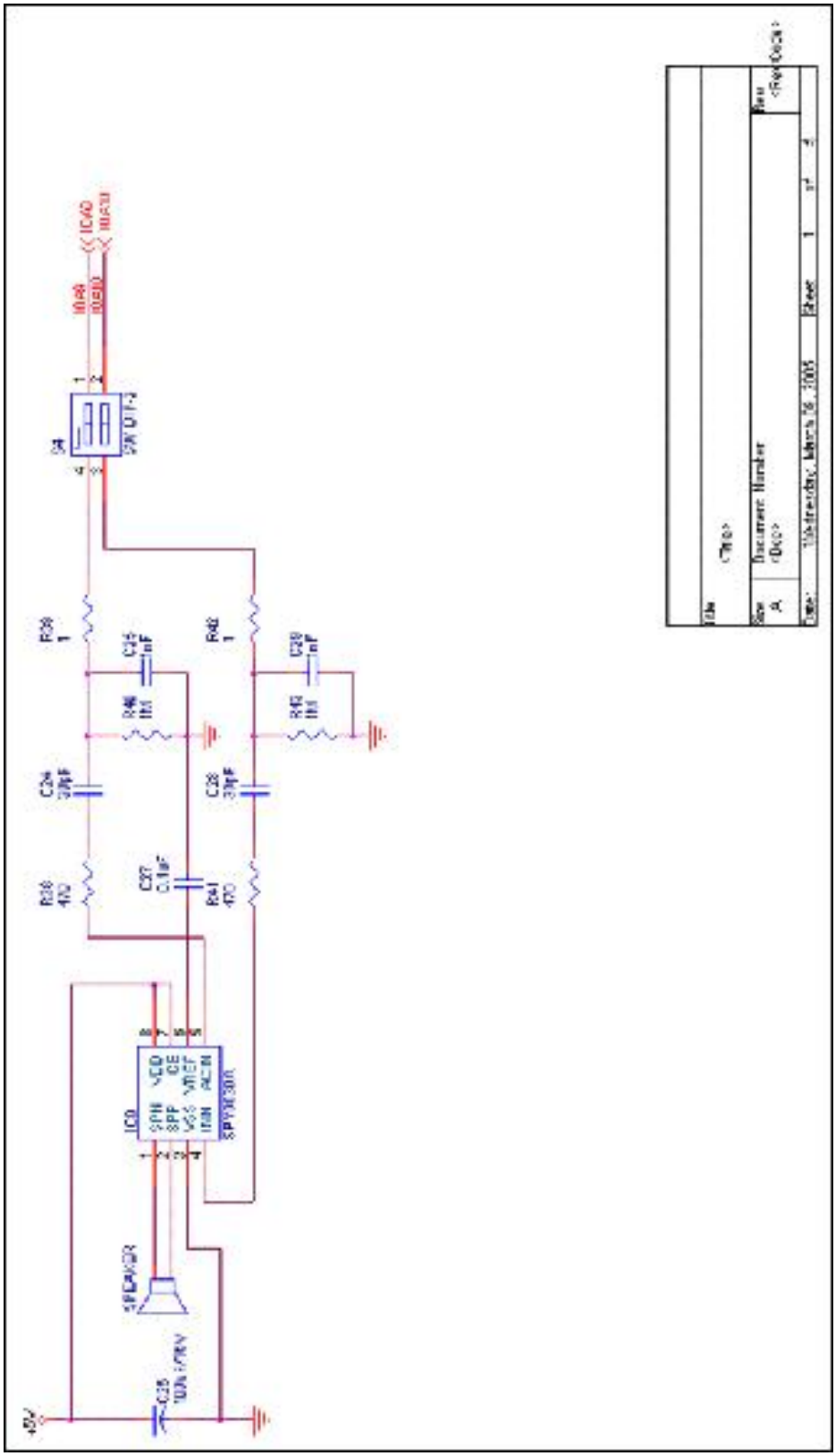


图22-4　SPMC75F2413A EVM 板原理- 4

第23章 凌阳创新 u'nSP 集成开发环境(IDE)

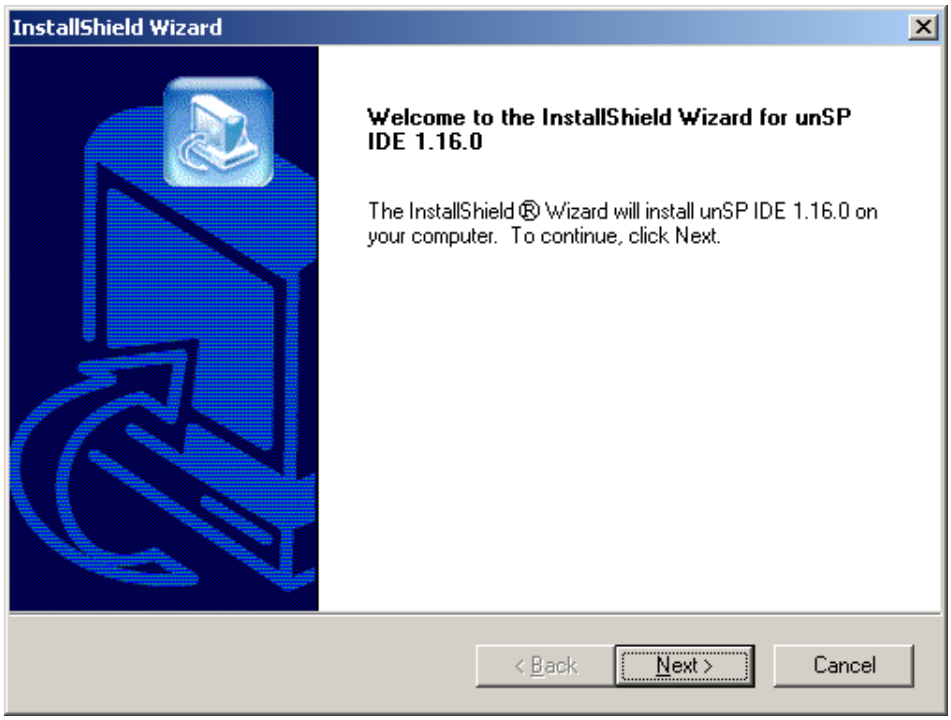
23.1 简介

欢迎使用凌阳创新 μ'nSP® IDE。μ'nSP® IDE 是凌阳创新 16 位 μ'nSP 内核的集成开发环境(IDE)，是基于 Windows 开发工具,集编辑、编译、链接、调试和仿真功能于一体。具有友好的交互界面、下拉菜单、快捷键和快速访问命令列表等，使编程、调试工作方便且高效。仿真模式也可以在不连接仿真板的情况下仿真程序。

23.2 安装

本节介绍如何安装 μ'nSP® IDE 到用户计算机上。μ'nSP® IDE 可以在 Windows98®, Windows2000® 和 WindowsXP®操作系统下运行。(如果在 Win2000/XP 操作系统下安装 μ'nSP® IDE，用户需要有网络管理权利在 BIOS 设置"SPP"模式打印口，推荐硬件地址为 378H. 如果用户使用 USB probe, 只需安装完后插上 USB probe)

运行按装程序，解压完成后，欢迎窗口显示如下：



1. 依据屏幕提示，完成安装
2. 选择 [Start Menu]→[Program]→[SunplusIT]→[unSPIDE]→[unSPIDE] 运行 IDE.
3. 安装完成后,用户若使用 USB probe, 插上 USB probe。Windows 系统将自动安装 USB probe 驱动程序
4. 在 Windows XP, 用户若使用 USB probe。插上 USB probe，Windows 系统将提示用户是否安装 USB probe 驱动程序，请选择[Continue(C)], 系统将自动安装 USB probe 驱动程序

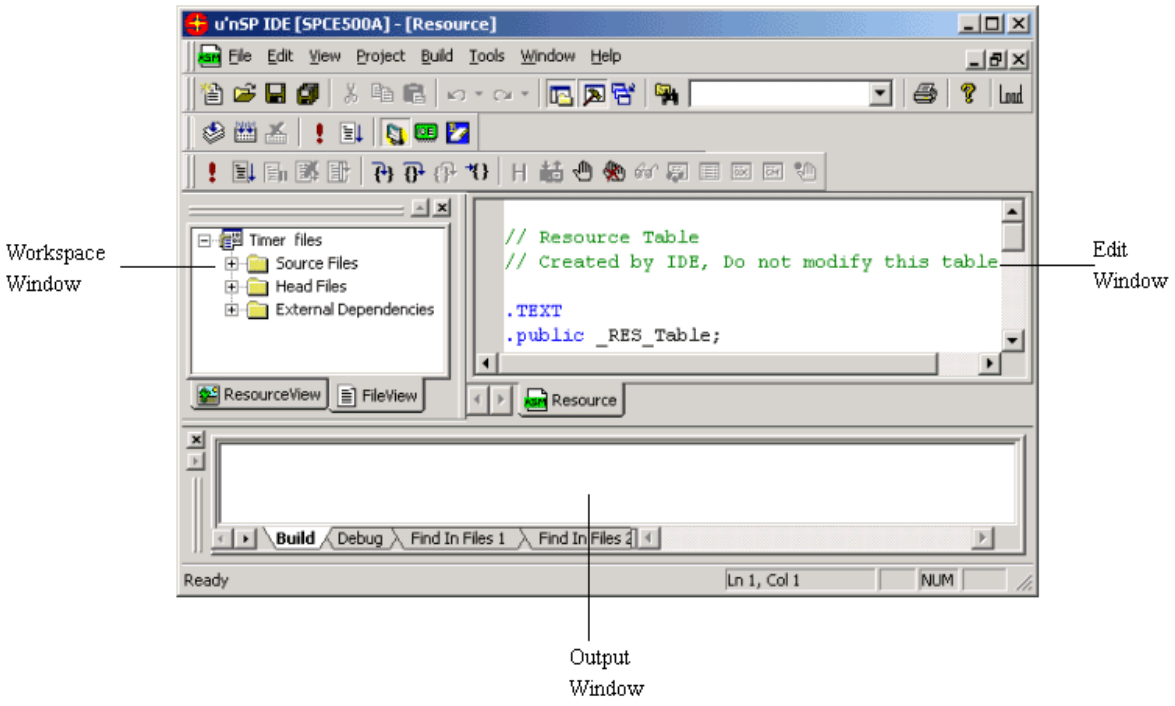
23.3 快速启动

1. 从开始菜单中运行 μ'nSP® IDE
2. 选择 [File]→[Open Project]
3. 工程窗口位于左边. 可以在 FileView 窗口双击文件名打开源程序
4. 选择[Build]→[[Rebuild All] 对源程序编译和连接。如果没有错误请进行下列步骤。
5. 选择[Build]→[Start Debug]→[Download] 装载程序，然后用 Debug 中的菜单命令运行/调试。用户可以选择[Build]→[Start Debug]→[Go]直接运行程序

23.4 μ'nSP® IDE 介绍

μ'nSP® IDE 包含了一系列集成开发工具，用于程序的编辑、编译、链接、调试和仿真。允许用户同时打开一个或多个窗口。μ'nSP® IDE 有三个主要的窗口工作区(Workspace)窗口，输出(Output)窗口和编辑(Edit)窗口。用户可以点击窗口内部区域以进行窗口之间的切换。

23.4.1 主窗口



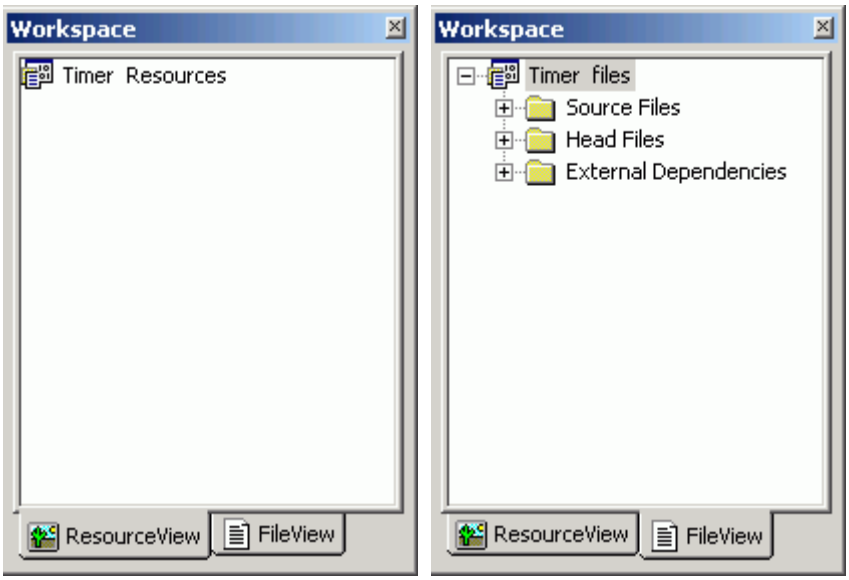
23.4.2 工作区(Workspace)窗口

工作区(Workspace) 显示文件信息。工作区窗口列出当前工程中所用文件及相互关系，称为 ResourceView 和 FileView。ResourceView 窗口列出当前工程中所用的全部资源文件。FileView 窗口列出当前工程中所有文件的逻辑关系,用户点击工作区窗口上的 ResourceView 或 FileView 按钮进行切换。

注：工作区窗口并不描述文件在硬盘上的物理位置；资源表是 IDE 依据资源树结构顺序创建；

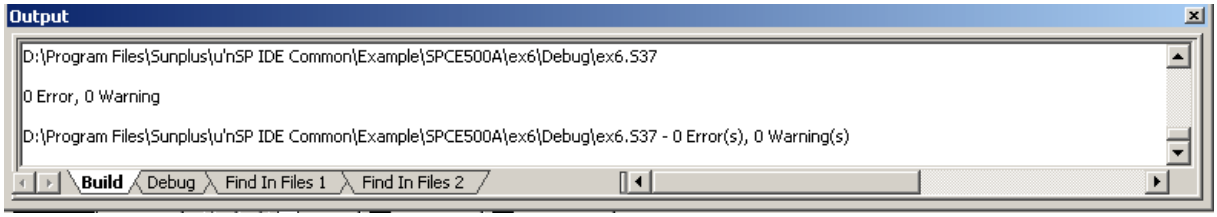
用户可通过鼠标的拖拉改变资源排列顺序。

〈ResourceView〉 〈FileView〉



23.4.3 输出（Output）窗口

输出(Output) 窗口显示编译，连接，调试，查找结果。

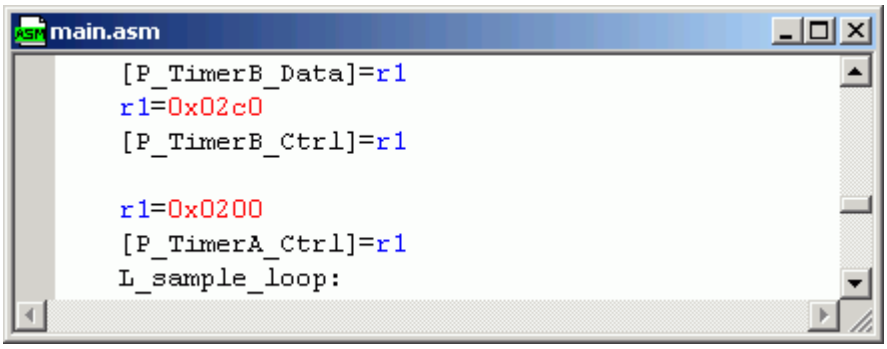


Build	显示编译，连接信息及编译，连接期间的错误或警告信息。程序编译，连接完,若没有错误或警告信息报告，就算成功。
调试:	显示调试信息。
在文件中查找	显示在文件中查找文本的查询结果。

23.4.4 编辑（Edit）窗口

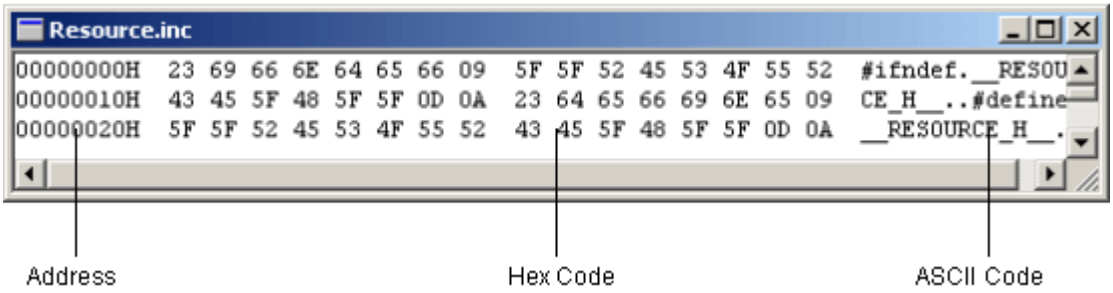
文本编辑器

用户可以用文本编辑器编辑程序。用户从工程中打开文件时，文件的内容会全部显示在文本编辑器中。



二进制编辑器

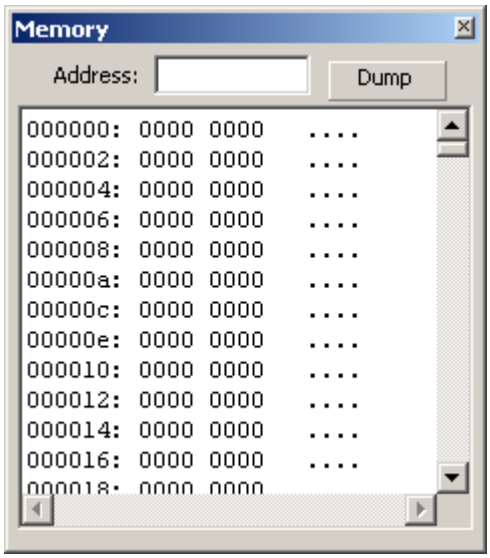
二进位制(Binary)编辑器用于以二进位制格式的十六进制或 ASCII 格式编辑现有的资源文件。



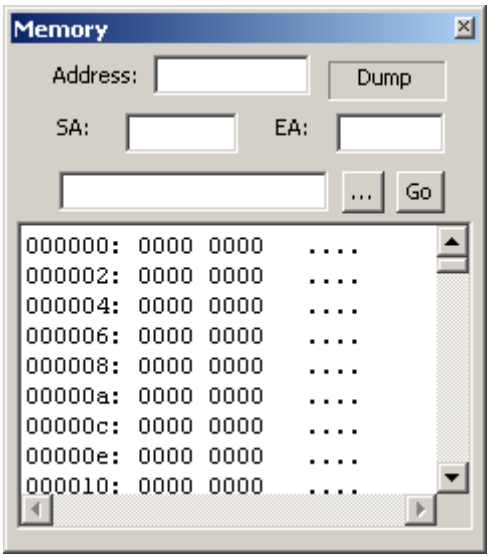
调试(DEBUG) 窗口

存储器窗口

调试时，用户通过选择 [View]→[Debug Windows]→[Memory]以查看存储器的内容。存储器窗口以十六进格式显示数据，允许用户在 Address 文本框中输入存储器的指定地址而查看其内容。那么存储器窗口显示指定起始地址的存储器内容，地址(Address)(默认 0x000000)。用户可以操作滚动条以查看存储器其它地址块的内容。此外，用户可以点击指定存储器单元，键入修改的值，以修改存储器指定单元的内容。



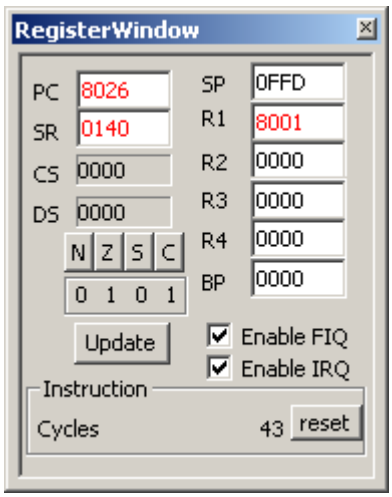
双击存储器窗口,以得到如下的"Go to Address"的对话框，键入要查看存储器地址，那么存储器窗口将显示从键入的指定地址起的存储器内容。



如上图所示，Dump 功能就是将指定区域 存储器的内容输出到一个文件，倘若，用户在 SA 和 EA 框中键入指定的存储器范围。然后点击[Go]以存储器的指定区域数据输出到指定文件。

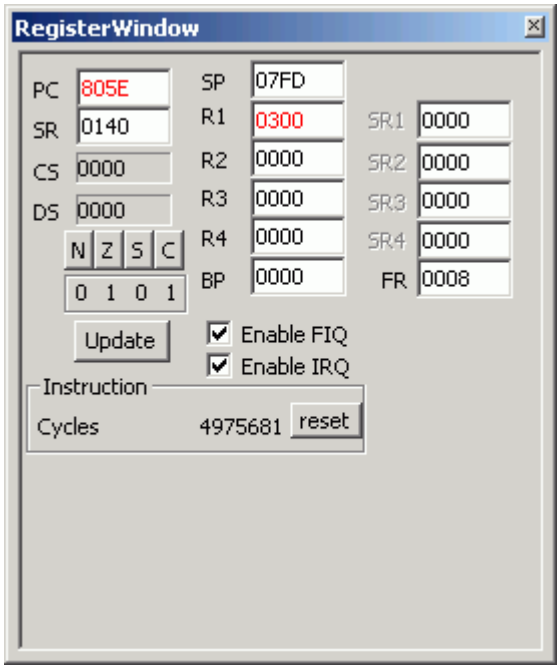
寄存器窗口

用户可以在这个窗口,对当前通用寄存器，CPU 状态寄存器进行选中和修改。

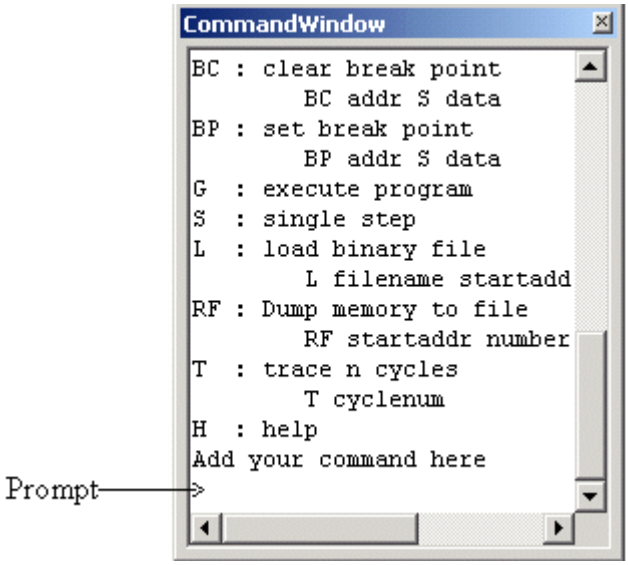


PC:	程序指针
SR:	状态寄存器
CS:	代码段
DS:	数据段
SP:	堆栈指针
R1 – R4:	通用寄存器
BP:	基指针
N, Z, S, C:	负数标志，零标志，符号位，进位标志
Cycles:	总周期数上次计数复位后到当前 PC 位置
Enable FIQ/IRQ:	使能或禁止 FIQ/IRQ
[Reset]:	配置复位
[Update]:	刷新用户对标志和中断使能的配置修改
SR1-SR4:	R1-R4 的二级寄存器
FR:	标志寄存器

注:在 ICE 模式下,用户无法看到指令周期计数器；ISA1.2 指令集下,增加了 5 个寄存器:



命令窗口



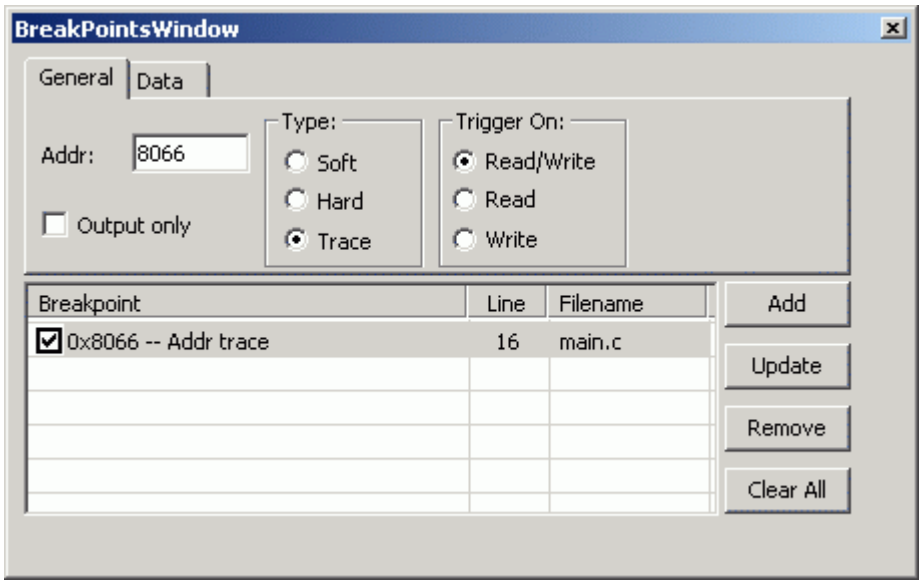
选择 [View]→[Debug Windows]→[CommandWindow] 以打开命令窗口。在命令文本对话框键入"H",在线帮助得以启动，下面列出全部命令和语法。用户可以在命令文本对话框窗口键入指定命令以执行命令。

命令	功能描述	语法格式及语例
Q	退出μ'nSP™ IDE	
Dump	转存存储器的内容	Dump <起始地址> <转储字数> Dump 100 100 //转储 0x100 - 0x1ff 中的字数据
EF	使能 FIQ 中断	

命令	功能描述	语法格式及语例
DF	禁止 FIQ 中断	
EI	使能 IRQ 中断	
DI	禁止 IRQ 中断	
SN	设置负标志	
NN	清除负标志	
SS	设置符号标志	
NS	清除符号标志	
SZ	设置零标志	
NZ	清除零标志	
SC	设置进位标志	
NC	清除进位标志	
X	复位（程序指针指向复位向量中的地址）	
RX	设定寄存器的值	Rx <寄存器号> <设定值> Rx 3 abcd //将 R3 的值设为 0xabcd
O	设定内存单元中的值	O <地址> <设定值> O 7016 abcd //将 0x7016 单元的值设为 abcd
F	设定 RAM 区中的值	F <起始地址> <结束地址> <设定值> F 100 1ff 1234 //将 0x100-0x1ff 单元填入 0x1234
BC	清除断点[1]	BC <断点地址> <断点标志> <断点数据> BC 8000 8082 1234 //清除当向 0x28000 单元中写入数据 0x1234 时的条件断点
BP	设置断点	BP <断点地址> <断点标志> <断点数据> BP 8000 8082 1234 //设置当向 0x28000 单元中写入数据 0x1234 时的条件断点
G	连续运行程序	
S	单步运行程序	
L	将二进制文件装入内存[2]	L <文件名> <起始地址> <结束地址> <起始地址> L test.bin 100 1ff 8000//将 test.bin 文件中第 0x100-0x1ff 单元的//数据装入 0x8000 单元
RF	将存储器内容转存到文件中	RF <起始地址> <单元个数> <文件名> RF 100 100 test.bin //将 0x100-0x1ff 单元的内容转储至 test.bin 文件中
H	显示命令帮助信息	显示 µ'n SP IDE 的所有命令及及其内容描述

断点(BREAKPOINT) 窗口

在断点窗口，用户可以设置，清除，使能，禁止和查看断点。断点设置将会作为工程的一部分而保存。

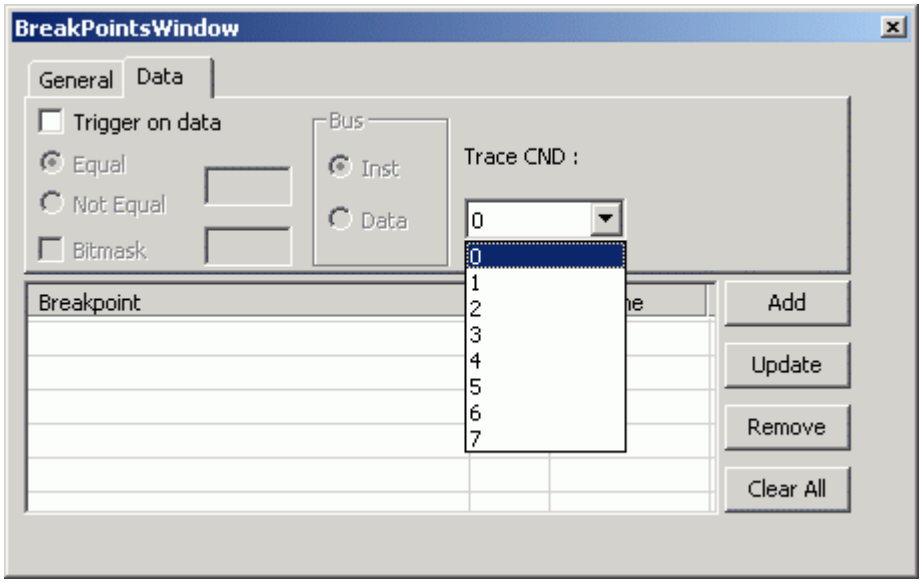


Addr: 断点的地址。

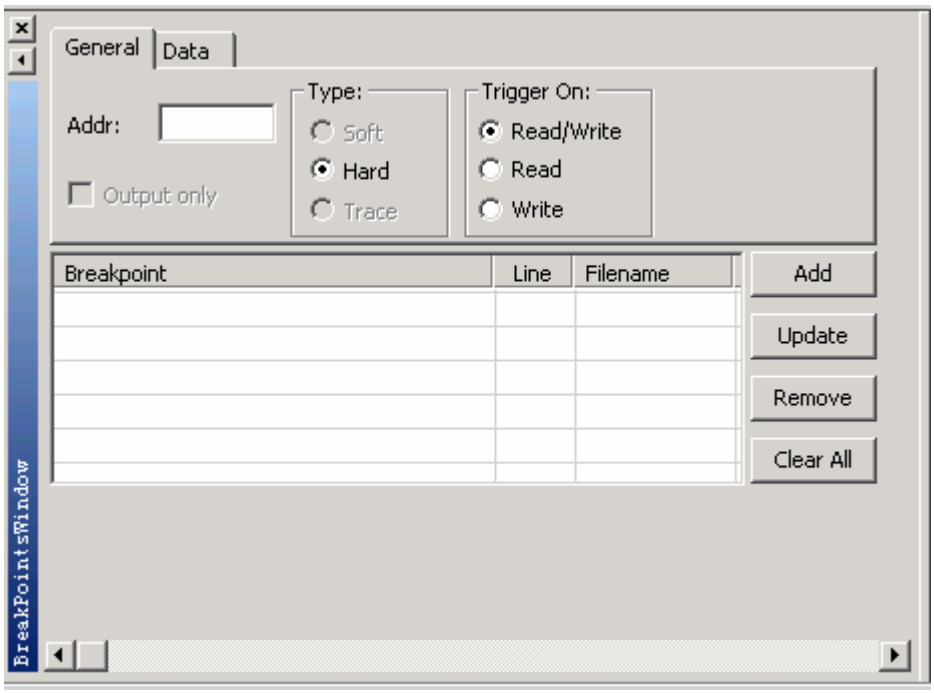
Type: 断点类型(设置).如果在 ICE 模式下，硬件条款使能，ISA 1.2 及以上版本软件条款使能，在仿真模式下，仅软件条款才能使能. 当追踪断点匹配时,从 tracebuffer 取出，程序将继续运行。

Output Only: 在 ICE 模式下芯片指定管脚输出脉冲信号。

Triggle on Write&Read: 当对指定址访问产生断点（中断）。



Trigger on Data: 在 ICE 模式下，如果用户在常规设置页选中 Hard(如下图)，以 Trigger on Data 使能。如果 Trigger on Data 和 Equal(Not Equal) 被选中，在运行中检测指定地址的数据是否和在 Triggle on Data 文本框中设定的数据值相等（不等），若匹配程序停止而进入断点。

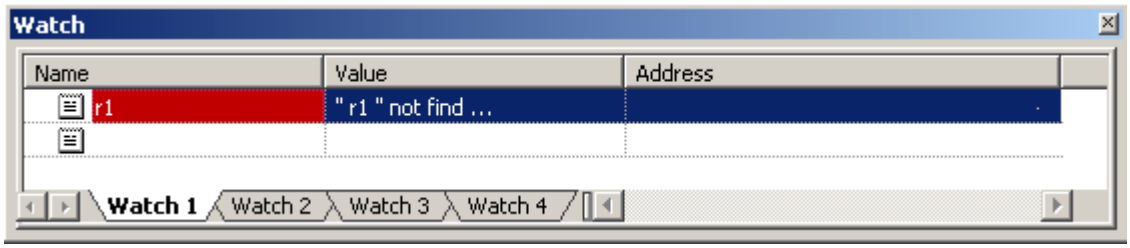


Bitmask: 如果 Bitmask 使能,那么 Not Equal 项使能,屏蔽指定地址数据的某些位。(以满足匹配后进入断点)

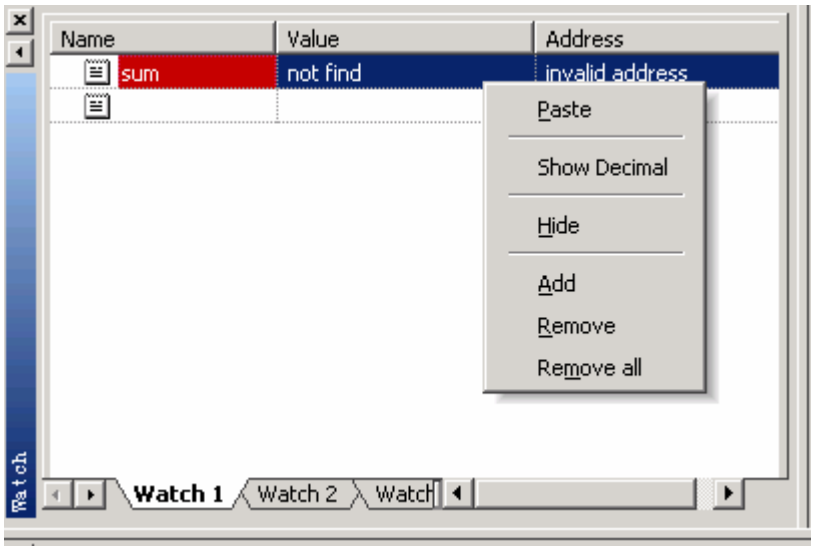
Bus: 在 ISA 1.2 及以上版本, 选择 Inst 设置指令类型, 选择设置数据类型。

Trace CND: 若选择 1-7, 相关的 CND 在程序运行期间将输出数据。

监视(WATCH) 窗口



监视窗口被用于在调试时对指定变量进行监视和修改。监视窗口包括 4 页表格: Watch1, Watch2, Watch3, 和 Watch4. 每页表格都能够以电子表格形式详细列出用户指定的变量, 变量若以红色显示, 则表明该变量的值在最近运行中发生了变化。右击该窗口内表格,会探出如下窗口。



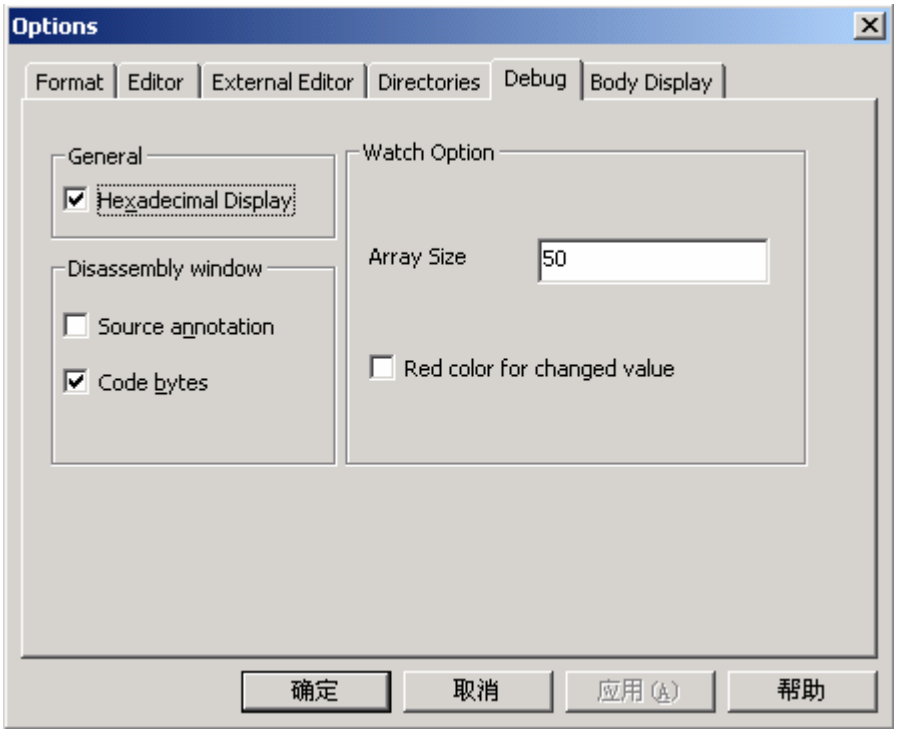
Paste: 粘贴

Show decimal/hexadecimal: 以进行十六进制/十进制显示变量

Hide: 隐藏监视窗口

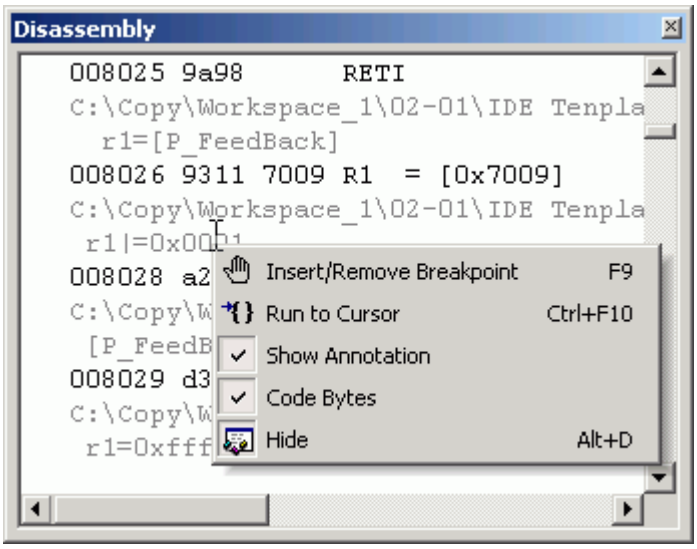
Add/Remove/Remove all: 在 WatchN(N=1,2,3,4)编辑当前列表(增加,清除,全部清除变量)

用户可以在[TOOL] →[Option]→[Debug](如下图) 中设置数组显示长度。若选中 Red for changed value,变量的值在发生变化其就为红色标明, 但会影响调试时速度。



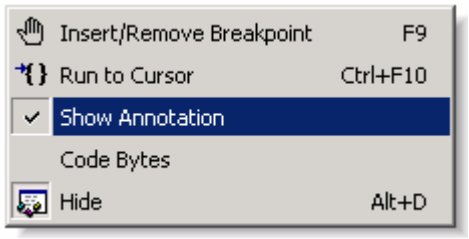
反汇编（DISASSEMBLY）窗口

反汇编窗口用于代码调试。其内容会随 PC 值而变化。

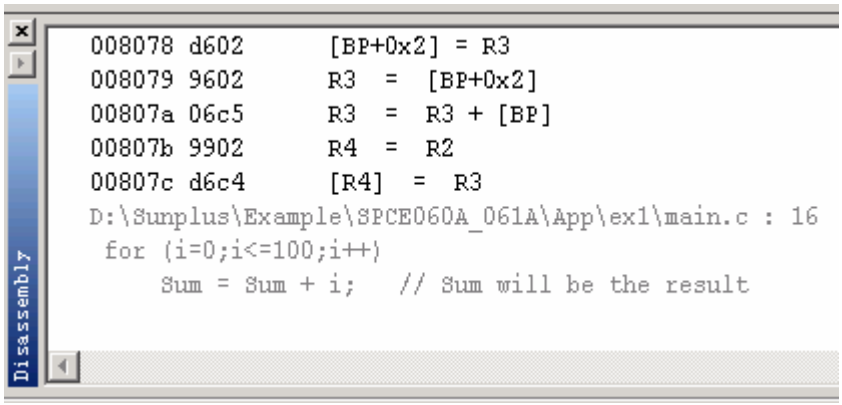


在反汇编窗口上双击，会弹出如下图所示对话框,用户可在对话框内键入指定地址，然后按[OK]，指定地址的反汇编会显示在 Disassembly 窗口内顶部显示。

单击右键，会弹出如下图所示下级的菜单。



用户可以通过选中[Show Annotation],然后右击选择评注,或选择[Tools]→[Option]→[Debug]→[Source annotation] 选择评注,如下图所示。

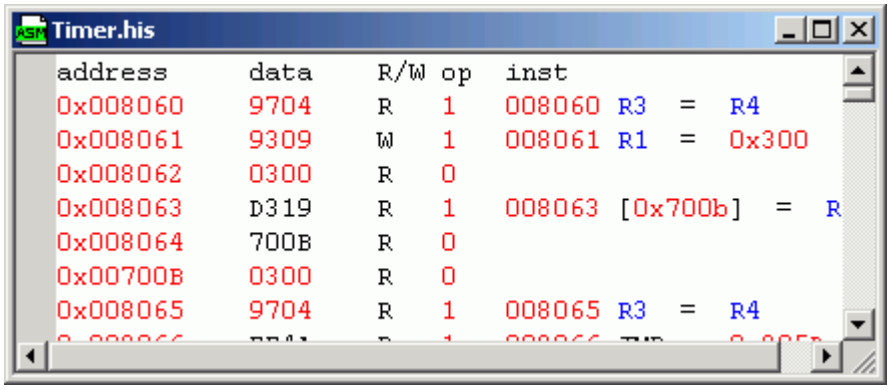


用户热键选择[Code Bytes]或选择[Tools]→[Option]→[Debug]→[Code bytes],以查看操作码；用户可以插入,清除断点,运行程序到光标处，查看源代码和操作码。

历史缓冲器窗口

程序运行时(当 TraceBuffer 在工程中设置使能)。程序的执行指令,状态和存储器内容会纪录在历史缓冲器中。在调试工具条中单击

，历史缓冲器窗口会显示如下:



如果用户在工程设置(project option)中选择"Save Instruction Only"，仅指令显示在窗口内。

23.5工程

23.5.1 工作区窗口

在工程窗口中，用户可以看到如下信息：

1. 源文件名称和其之间的逻辑关系。在工程窗口中,文件之间是目录树的逻辑关系。并不表明物理意义上的硬盘位置。 双击源文件名称，会在编辑窗口中打开。
2. 当前程序的工具配置，如，编译器和连接器的选项。选择[Project]→[Setting] 来设

置工程选项。

工程的操作

用户可以在工程中添加或移去文件或刷新当前工程中文之间的逻辑关系。

工程内各类文件

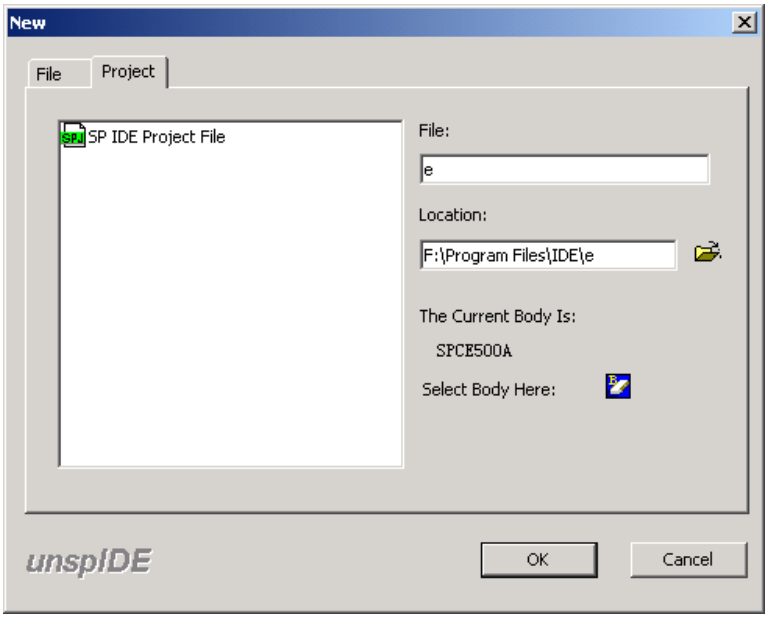
一个工程包含 5 个类型文件:

1. 工程文件 (.spj): 扩展名 'spj', 其包含创建工程的命令和设置
2. 资源文件 (.rc): 扩展名'rc', 其包含当前工程中所有资源文件
3. 资源列表文件(.asm): 资源表是依据资源的树型顺序够建的
4. 资源表头文件(.inc)
5. 构造文件(Makefile)
6. C 包含文件(.h)

注: 用户不能直接修改.spj 和 .rc 文件

创建工程

选择 [File]→[New] 显示一个 "New" 对话框。



1. 单击 Project 条
2. 在文件框键入工程名
3. 浏览和选择路径, 直接输入工程的指定路径
4. 选择 body 类型
5. 单击 [OK]

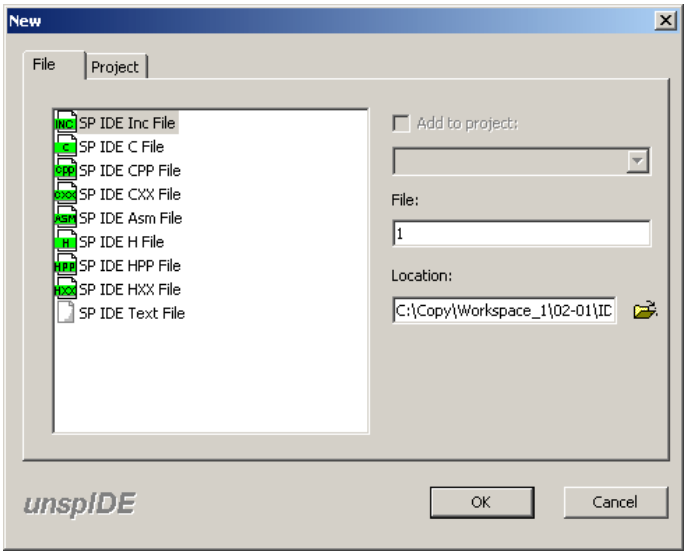
这时，μ'nSP® IDE 生成一个新工程，工程信息在 Workspace 窗口内显示。如果 Workspace 窗口没有显现，可以选择[View]→[Workspace]，打开 Workspace 窗口。

创建工程时， μ'nSP® IDE 会自动创建如下文件:.spj(project file) .rc (resource file), .h file (C

included file), Resource.asm (Resource Table), Resource.inc (Resource Table Head file), 和 Makefile.例如 创建一个"test"工程, μ'nSP® IDE 会自动在"test" 路径下生成 "test.rc" 文件, test.spj, Resource.asm, Resource.inc, Makefile and a "test.h"文件。

新建源文件

选择[File]→[New]，打开 New 对话框：



1. 在 File 页的文件类型清单里选择 SP IDE C File;
2. 在 File 框内输入源文件名称，Location 框内包含了生成文件时指定的工程文件夹路径;
3. 选择 [OK]。

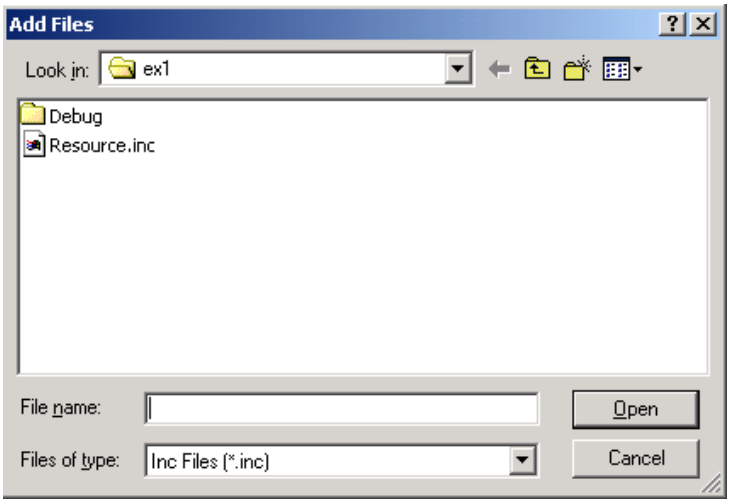
选择 Add to project，可把源文件添加到某个工程内。

输入程序代码后，选择[File]→[Save]，可保存输入的内容。

向工程添加文件和资源

可通过两种办法向工程添加文件：

1. 选择[Project]→[Add to Project]→[Files/Resource]，激活 Add Files 对话框。
2. 在工程窗口的文件夹上单击鼠标右键，在热键菜单里选择 Add File To Folder，激活 Add Files 对话框。



2. 在文件列表中选择一个或多个。
3. 单击 [Open]。

打开工程

当打开一个已有工程, μ'nSP® IDE 自动恢复上次设置。打开工程有三种方法：

1. 第一种打开一个已有工程方法：
   （1） 择 [File]→[Open Project] 显示"Open" 对话框。
   （2） 文件类型列表中选择 'Project Files(*.spj)/(*.scs)/Pack File(*.pak)。
2. 第二种打开一个已有工程方法：
   （1）选择[File]→[Recent Projects]，列出的最近打开过的 4 个工程。
3. 第三种打开一个已有工程方法：
   （1） 择 [File]→[Open] 显示"Open" 对话框。
   （2） 文件类型列表中选择'Project Files(*.spj)。
   （3） 工程列表中选择一个工程,选择打开类型为 text. 然后单击 [Open] 或直接双击所选工程。

   注: 单击 "Preview" 可以在 Open 对话框预览文本文件。

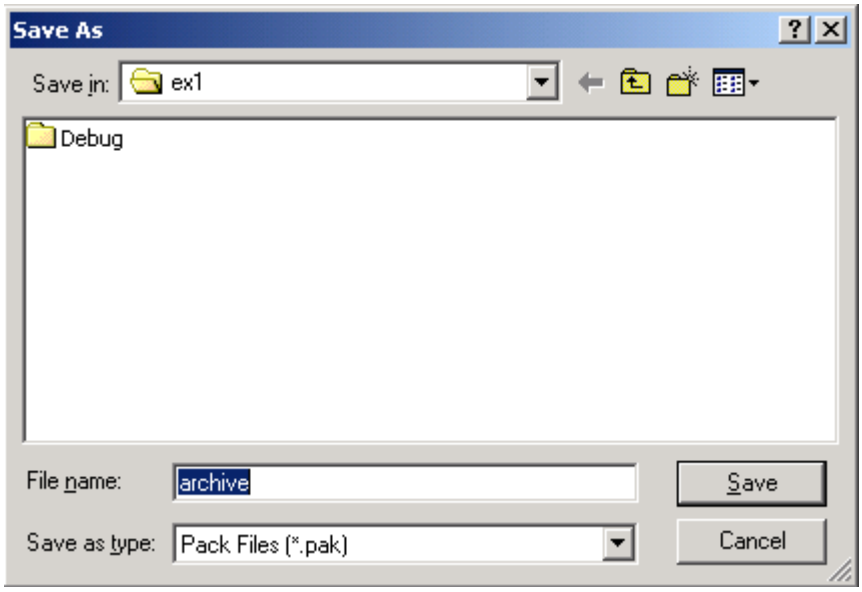
保存工程

1. 选择 [File]→[Save Project] 用来保存工程和相关配置。
2. 用户 rebuild 工程,关闭时, 系统会自动保存工程。

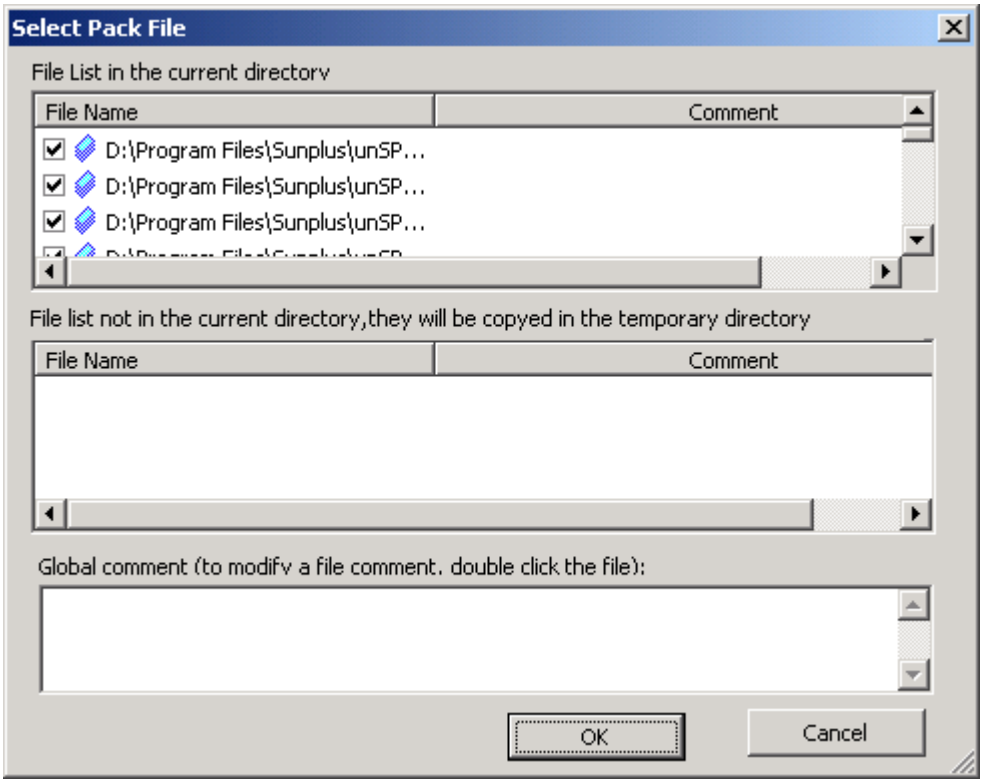
封装工程

封装工程的功能可以帮助用户包装当前工程的全部信息到一个.pak 文件, 任何时候，用户打开 pack 文件都可恢复工程的配置。

1. 创建 Pack 文件
   (1) 选择 [File]→[Pack Project], 会弹出“Save As”。的对话框，如下图。



(2) 在文件名框中输入包装文件的名,然后单击[Save]。单击[Save]后, 系统弹出一个 Select Pack File 对话框:

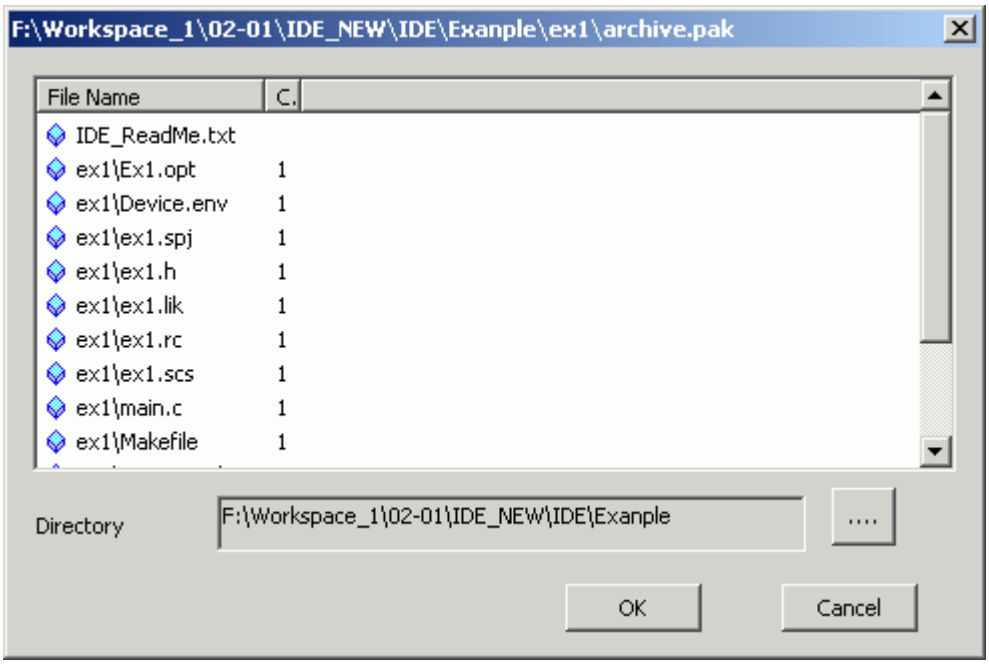


该窗口前两个列表内，可选择需要封装的文件。所有非本工程路径内的文件在进行工程封装后将被拷贝到临时目录内。在 Global comment 框内，用户可指定所有文件的注解。当然，用户可为单个文件写注解，方法为双击前两个列表框内的文件名称，然后，在文本框内写注解。选择 [OK]，生成 pak 文件。

(3)单击 [OK]以产生 .pak 文件。

2. 解包 Pack 文件

选择 [File]→[Open Project] 打开 .pak 文件。先好存贮径后点击[OK]就可。



创建目标文件

内部目标文件是一个当前工程的二进制输出文件。 μ'nSP® IDE 依据源文件和工程描述创建一个目标文件。通常, μ'nSP® IDE 依照下面流程创建一个目标文件:

1. 二进制输出文件的格式(格式 S37, TSK)
2. 设置编译器和连接器

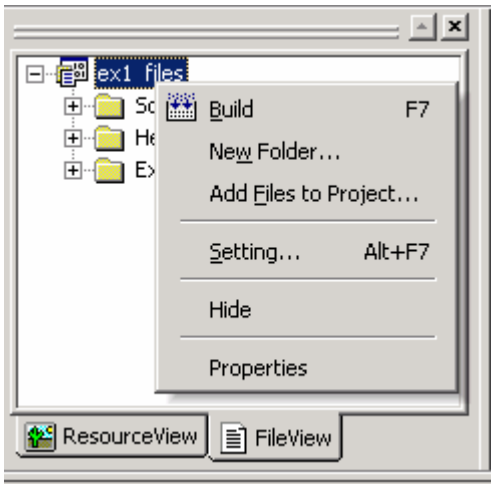
用户可以在任何时候通过 build 的下拉菜单选择创建一个新的调试(Debug) /发行(Release)目标。

工程中的文件夹

创建一个工程后，μ'nSP® IDE 会自动在工程中创建三个文件夹 (Source Files, Head Files and External Dependencies) ,如果需要的话，用户也可以创建额外的用于保存文件的文件夹，文件夹可以显示文件间的逻辑关系，例如，用户可以创建一个包含所有用户接口文件或其他如工程描述，记录文件，测试文件组等的文件夹。所有文件具有不同图标

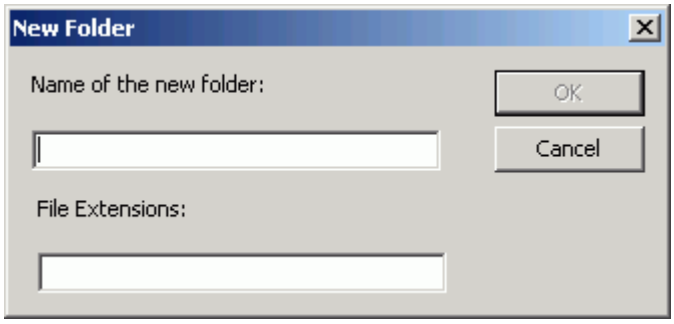
添加文件夹到工程

1. 在工程窗口，将光标移至工程或文件夹上，然后右击如下图所示。



2. 选择菜单上的"New Folder"。

3. 弹出如下新文件夹窗口对话框，在新文件夹名框（Name of the new folder ）输入文件夹名称。

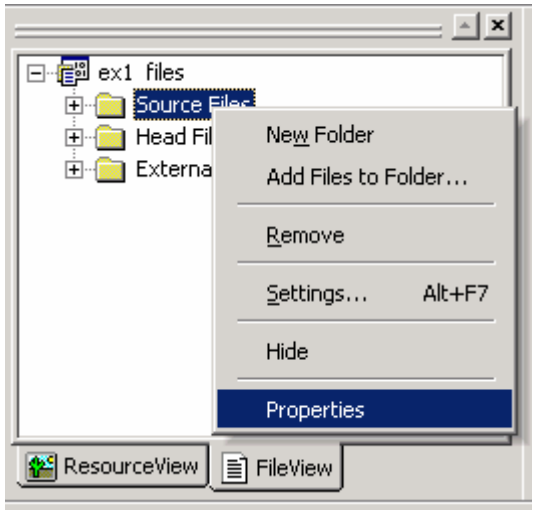


4.在上面框中输入文件名，然后在文件扩展名框（File Extensions ）输入扩展名。

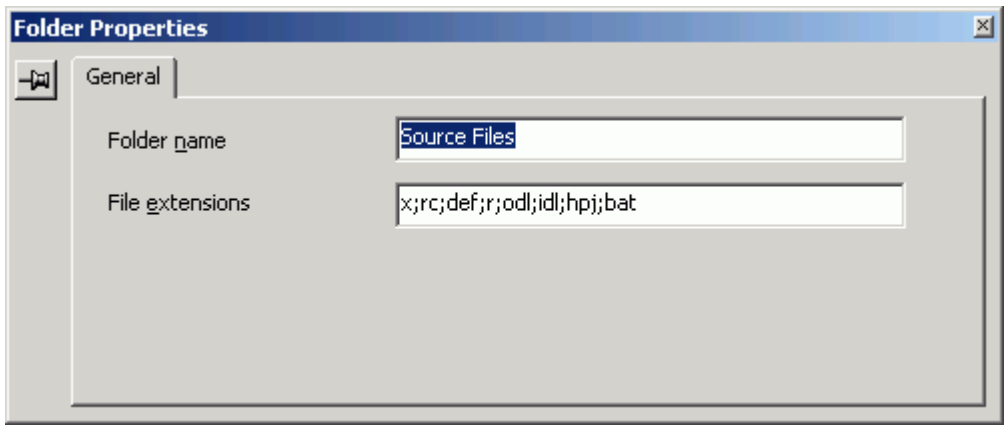
5. 设置文件夹以用于添加或清除资源文件。

改变文件夹属性(CHANGE GROUP PROPERTY)

在 FileView 窗口改变文件夹属性，在 FileView 选择文件夹，然后右击，选择"Properties"，如下图所示：

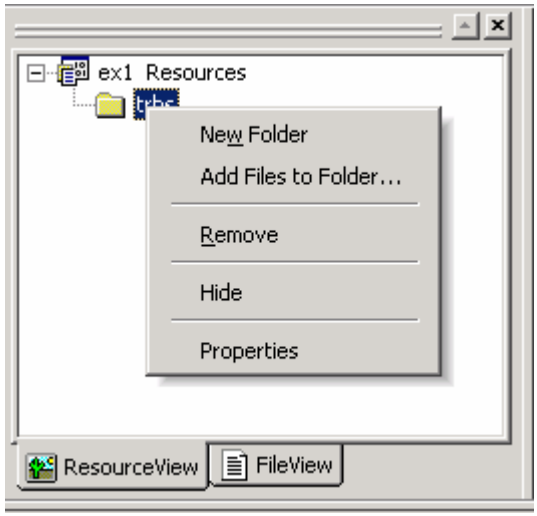


选择 "Properties"选择 然后右击,如下图所示：

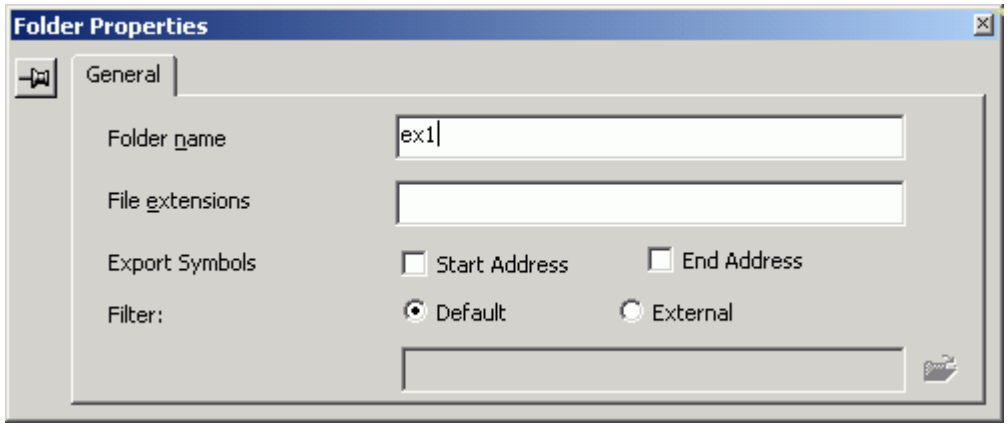


在 ResourceView 窗口中改变文件夹属性, 选择 ResourceView 窗口然后右击,选择 "Properties",如下：

选择 ResourceView 窗口然后右击



选择 "Properties"选择 然后右击,如下图所示



当前文件夹名称和文件扩展名称通过设置 Folder name 和 File extensions。文件夹输出标志和过滤器通过设置 Export Symbols 和 Filter。连接前, 用户可以选择 filter 为

external 将资源文件转化为凌阳创新格式。

外部过滤器的格式：

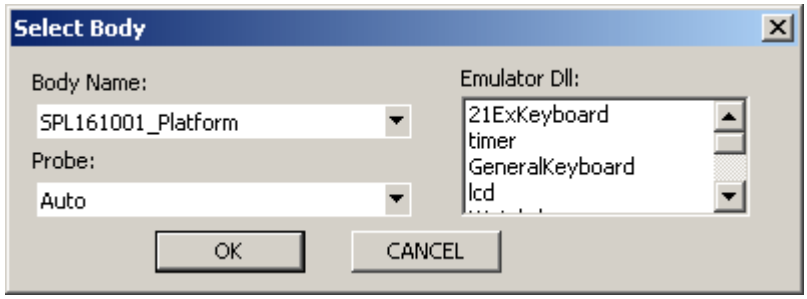
Filter.exe <input file> <*output.sp> *输出扩展名为 ".sp"

文件和文件夹的移动(MOVE FILE AND GROUP)

1. 在工程窗口中选择一个或多个文件或文件夹
2. 拖拉选择的文件或群组到工程或指定文件夹
3. 如果要移动文件，那么指定文件夹(类型)匹配于这些移动的资源文件

选择 BODY

选择 [Project]→[Select Body]以选择器件的 body 类型。



Body Name： 选择 body。

Probe： 选择连接 ICE 的 probe 的类型。

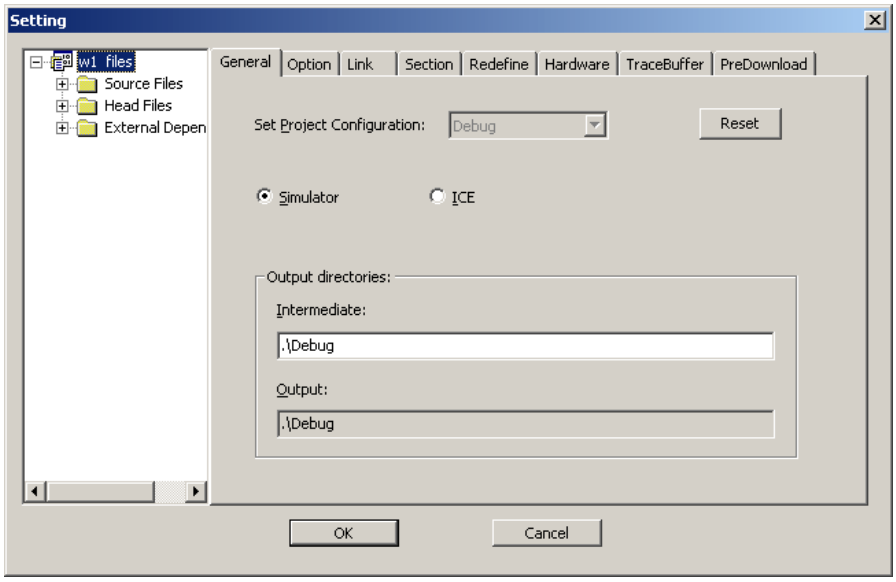
Emulator Dll： 选择仿真的动态连接库。

工程选项(SET OPTION FOR PROJECT)

对大多数工程来说，选择设置应在工程级。用户可以配置由 system/user 创建不同的目标，通常来说 μ'nSP® IDE 将创建两种目标：调试或发行。工程可以显示对每个目标的设置架构。设　置可以是工程级也可适应工程中所有文件。

1. 打开或创建一个工程。
2. 选择[Project]→[Setting] 将显示 "Setting" 对话框。
3. 单击每个标签做下一级设置。

[General]:



Set Project Configuration：显示工程使用的版本。

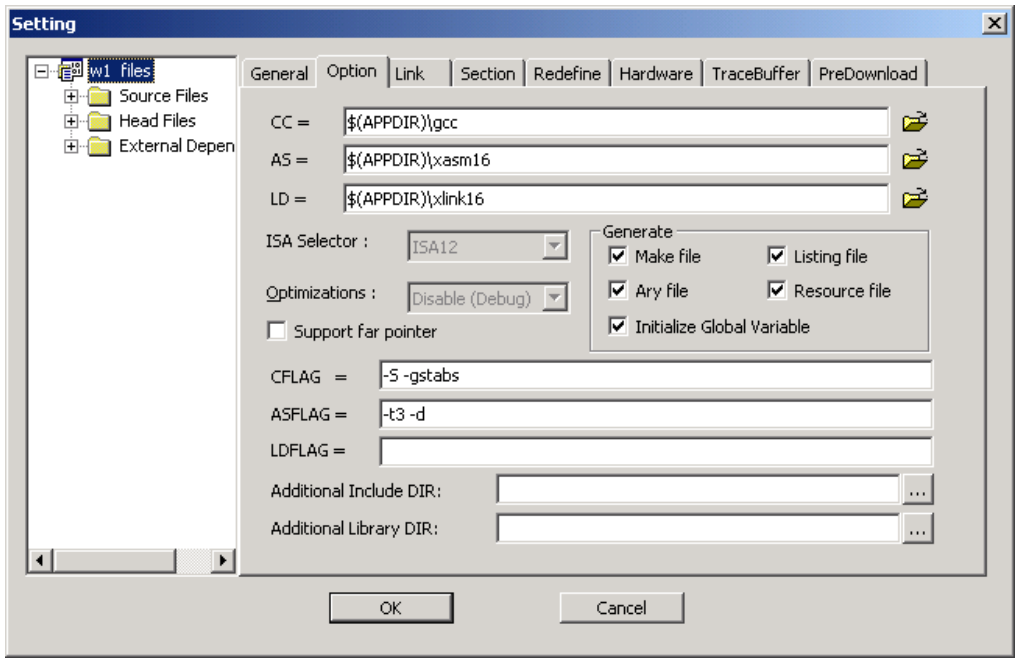
Simulator/ICE：选择 μ'nSP® IDE 运行模式。如果选择 ICE 运行模式，用户应将仿真板和计算机通过并口相连。如果选择模拟器，那么所有数据会存放于缓冲器中。

Intermediate：为中间文件设定目录. 中间文件通常在编译时产生。

Output：显示最终文件的目录. 通常,和中间目录相同。

Reset：恢复默认配置。

[Option]:



CC：指定 C 编译器位置

AS：指定汇编器位置

LD：指定连接器位置

CFLAG：指定 C 编译器标志（FLAGS）

ASFLAG：指定汇编器标志（FLAGS）

LDFLAG：指定连接器标志（FLAGS）

Optimizations：选择用户所需优化类型，优化标志将自动改变

ISA Selector：显示指令集

Support far pointer：选中则支持长指针

注：如果“支持长指针”，“CC”将被指定为“xgcc”，连接“Cmacro1x32.lib”。否则，“CC”将被指定为“gcc”，连接“Cmacro1x16.lib”

Makefile：选中该项以自动更新构造文件

Listing file：选中该项以生成列表文件

Ary file：选中该项以自动更新排列文件

Resource file：选中该项以生成资源文件(resource.inc and resource.asm)

Initialize Global Variable：选中该项以初始化全局变量

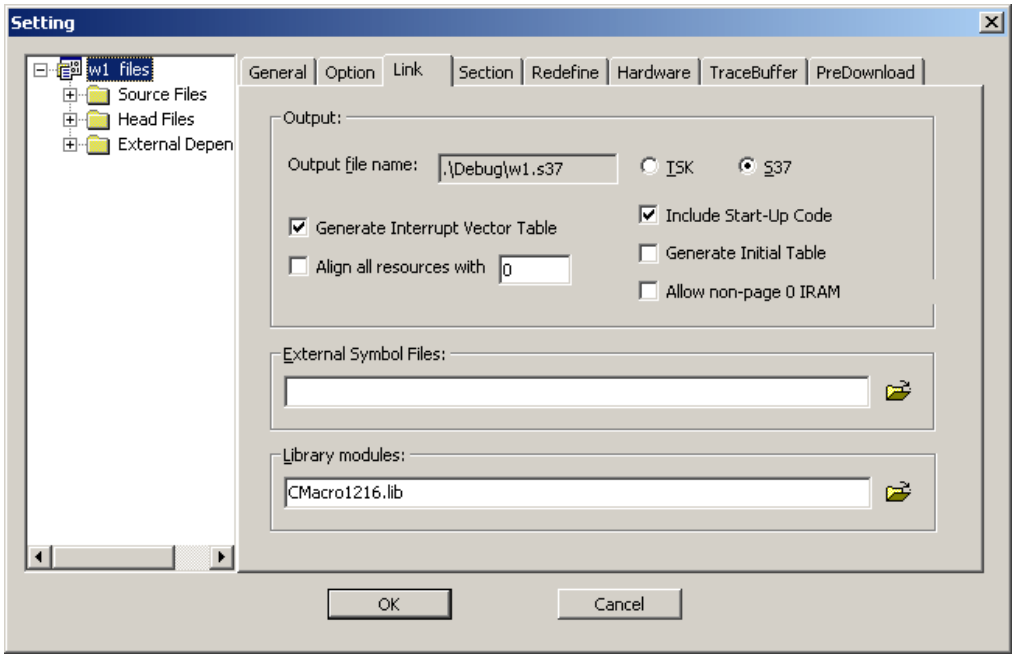
Additional Include Dir：指定包含文件的目录

Additional Library Dir：指定库文件的目录

在调试和发行模式里,CFLAG, ASFLAG, LDFLAG 可被指定为不同的参数。当在[General]中改变 build 模式，参数自动转换正确一边。

用户可以在“LDFLAG”中设定“-tsklen xxxx”以限制输出 TSK 长度. 例如,用户若欲限制 tsk 长度为 0x6ffff，那就设置为“-tsklen 6ffff”。

[Link]:



Output file name：指定输出文件名

TSK/S37：目标文件的类型（binary/ASCII（Motorola S37））. 在选择目标文件类型前，应在 Option 页选中 Makefile 和 Ary file

Generate Interrupt Vector Table：不选中，用户工程输出就无法连接生成中断向量表

Include Start-Up Code：不选中，用户工程输出就无法生成默认的引导代码（start-up code）

Align all resource with：如果用户需要依据基础排列，排列所有资源，就需选中且输入基础排列

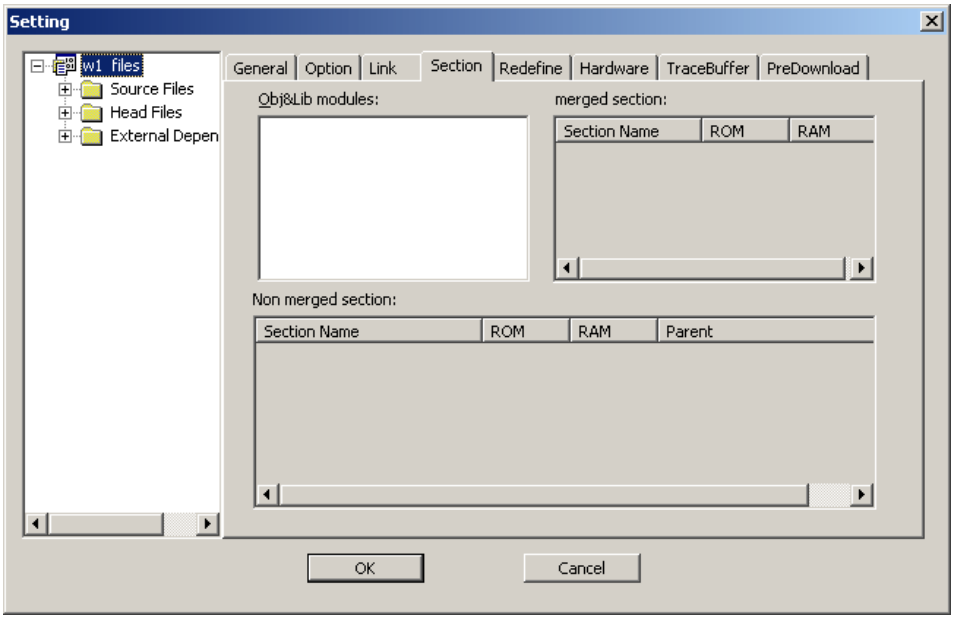
Generate Initial Table：如果用户在工程输出中不需要一个初始表，就不选中

External Symbol Files：输入其他符号文件作为当前工程符号文件

Allow non-page 0 IRAM：使能 IRAM 占用非零页

Library modules：指定并显示包含在当前工程中的库模块

[Section]:

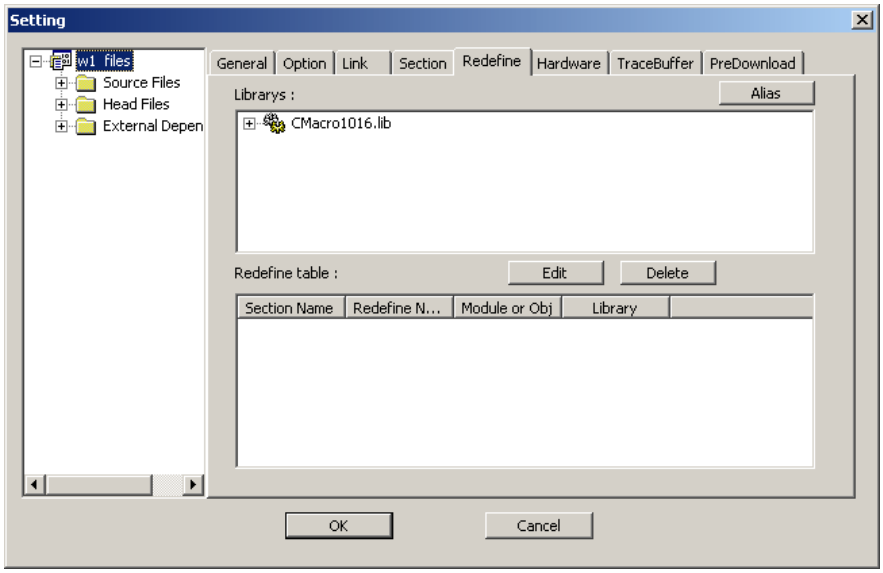


Obj & Lib modules：显示当前工程中所有的目标和库模块

Emerge section：列出当前工程中的 emerged(位置指定) 段

Non-emerged section：列出当前工程中的非 emerged 段。用户可以通过双击 ROM 区改变这些段的地址或排列基础. 重新连接工程后设定段和基础排列被锁定合适地址

[Redefine]:

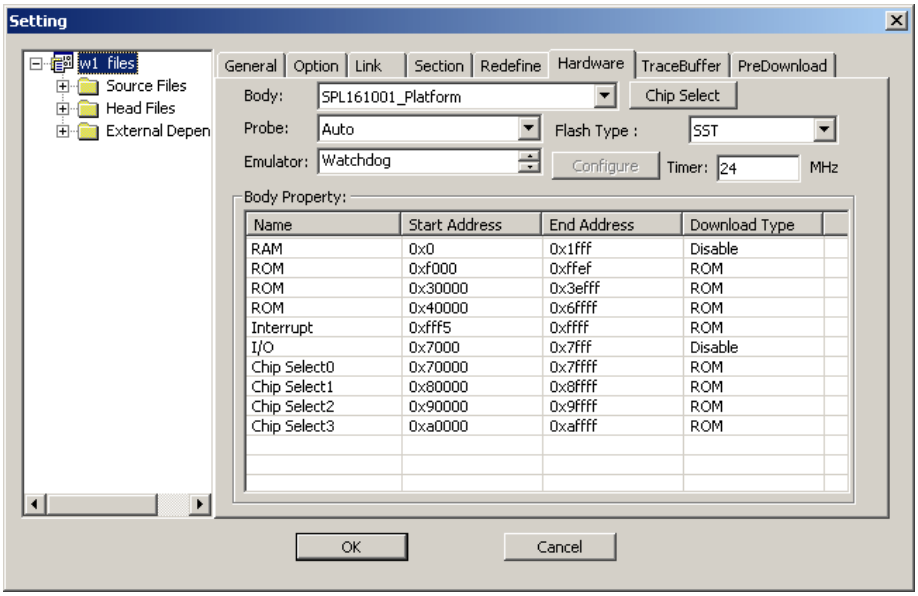


Alias: 从库列单窗口中选择段，然后重新定义为其他名称

Edit: 编辑在重新定义列单窗口中选中的条目。等同双击重新定义表单

Delete: 删除在重新定义列单窗口中选中的条目

[Hardware]:



提供芯片 body 选择 和信息区(Information Block) 设置.

Body: 选择 body。Body 的描述是连接器和模拟器连接和模拟的基础.

Probe: 选择连接 ICE 的 probe 的类型.

Emulator: 根据 IC 型号选择外部仿真器, 仿真器从 cpt 文件里的动态链接库指定匹配的 body.

Timer: 为仿真器和模拟器选择仿真时钟周期.

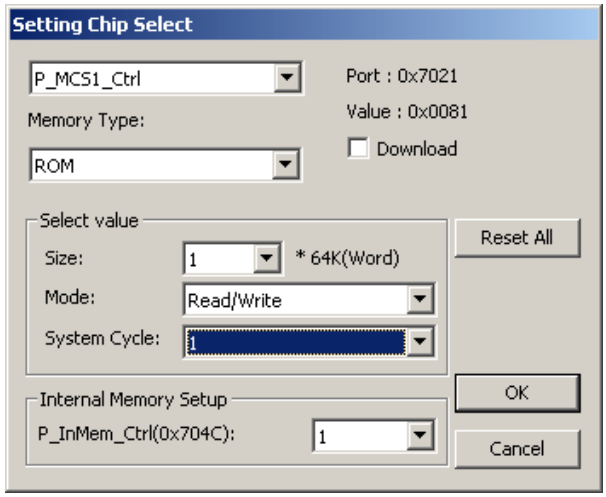
Configure: 设置仿真器.

Chip Select: 芯片选择.

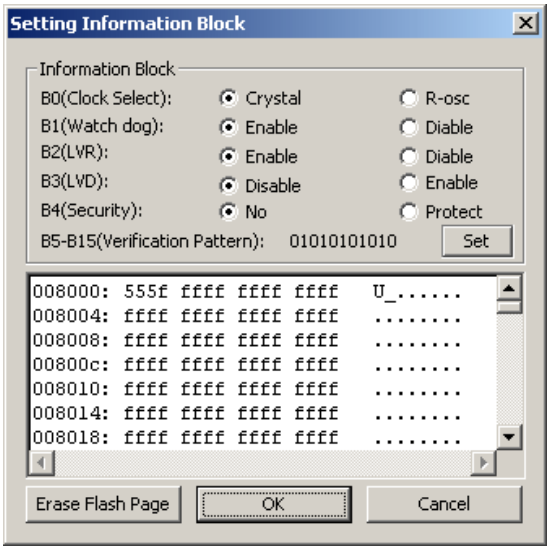
Body property: 显示存储器 mapping.

Flash Type: 用户选择支持 Flash 的 body,然后选择 Flash 类型.

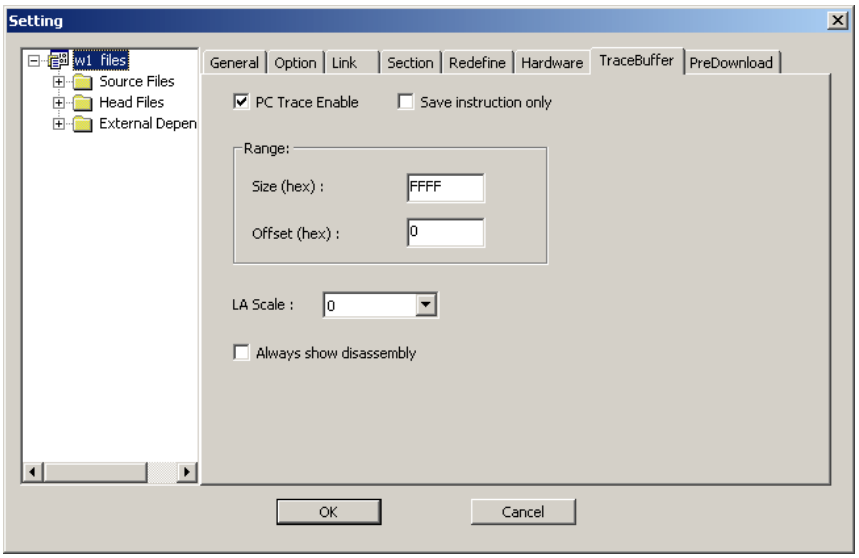
Set chip select:



Set information block:



[TraceBuffer]:



PC Trace Enable：PC 追踪功能使能(在 ICE 模式下,请确认您的 ice board 具有 PC 追踪功能)

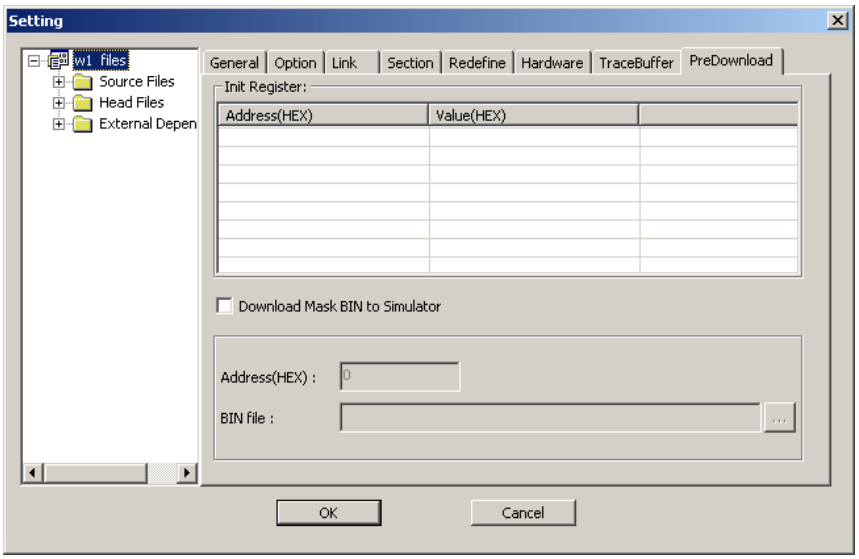
Save Instruction Only:如果选中 only instruction 指令将被存到 pc 的 trace buffer,否则存储所有存储器读/写到 pc 的 trace buffer

LA Scale：逻辑分析仪刻度配置

Size & Offset：设置缓冲器的大小和偏移量

Always show disassembly：如果选中,反汇编窗口总显示在主界面

[PreDownload]:

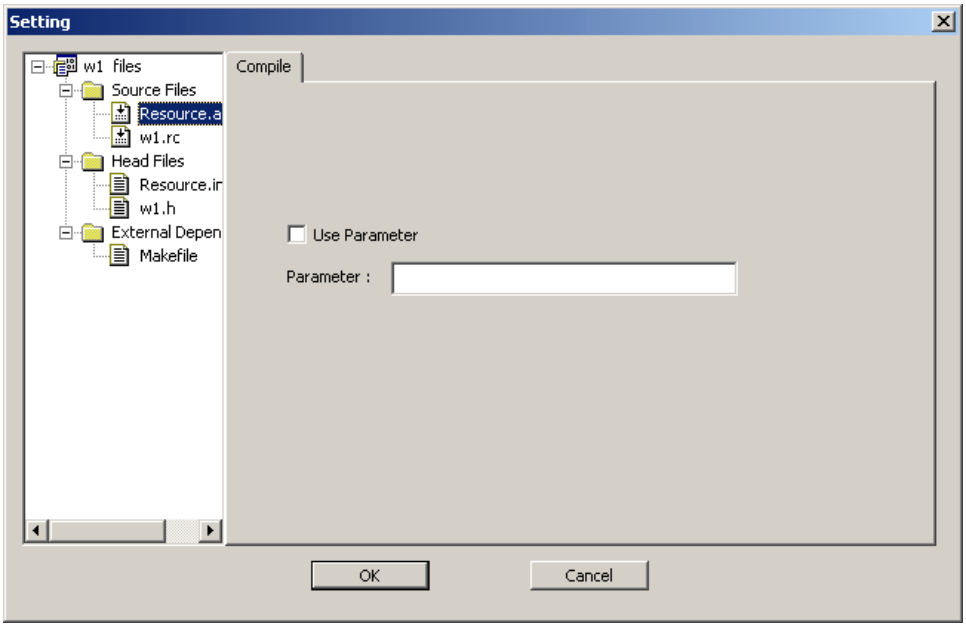


Init Register: ICE 模式下,下载之前写数据到存储器

Download Mask BIN to Simulator: 在仿真模式下, 下载之前写数据到存储器. 它可以从 'Address' 开始 数据来自 'BIN file'

设置文件选项：

用户可以为编译器设定参数。依此做法，选择 [Project]→[Setting...] 使能配置对话框，然后从左边目录树选择文件，显示编译设置页如下。



选择 "Use Parameter" 输入参数到文本对话框。有效设置如下：

-S: 编译到汇编语言

-E: 仅运行 C 程序的预处理部分

-o file：输出到'file'文件

--help：显示 gcc 命令行的参数的帮助

-ansi: 完全支持标准 ANSI C 程序。关掉 GCC 和标准 ANSI C 不兼容部分的特性。'-ansi' 选项不影响(无理拒绝)非 ANSI 程序. 因而，'-pedantic' 添加到 '-ansi'

-pedantic：严格依据 ANSI C 和 ISO C++ 发布必要的警告

-w: 抑制所有警告消息

-Wall: 使能所有用户认为可疑的指令，这很容易避免（或修改为预防警告），甚至有宏的关联

-Werror：使所用警告为错误

-Q: 使编译器显示被编译的所用函数的名称，在 _nishes 时，显示一些关于每次(编译)数据统计

-Dmacro: 定义大量的用字串'1'定义的宏. -Dmacro=defn: 定义大量的宏为 'defn'. 所有的'-D'在命令行上的场合在任何 '-U' 选项之前处理

-Umacro：不定义大量宏

-gstab: 为调试器产生信息

-Idir: 添加目录'dir'到头文件搜寻的目录列表的头(文件)

-OO：不优化

-O1：编译器尽量缩减代码大小和执行时间

-O2：比 -O1 更深的优化. 几乎支持不包括 space-speed tradeoff (速度和代码折中) 执行的所有优化

-O3：比 -O2 更深的优化. 包含-O2 所有优化. 并附加 , in-lining(内嵌汇编)功能

-Os：优化代码大小. 使能所有 -O2 optimizations 典型的代码大小优化. 还执行更深缩小代码的优化设计

BUILD 工程(BUILD PROJECT)

Use μ'nSP® IDE build 和 rebuild 工程中定义程序和库。build 一个工程, μ'nSP® IDE 仅处理上次编译之后而又编辑过的文件。对整个工程进行 build , μ'nSP® IDE rebuild 当前工程和其相关配置。

通常, μ'nSP® IDE 为新创建的工程提供一个默认配置. 调试目标包含足够的调试符号用于 μ'nSP® IDE 调试器调试信息, 所有的优化 选项在调试中无效,因为这些选项将妨碍调试。发行目标没调试信息符号, 在创建一个目标优化选项是有效的。

每个目标都为中间文件和输出文件指定路径。

Build Current Object

请从 μ'nSP® IDE [Help]→[About SunplusITIDE]注册。注册后, 用户才可以编译,连接,调试工程。

1. 选择 [Build]→[Compile] 编译编辑窗口中当前激活的文件。(等同于 右击 *.asm, *.c, 等. 在 Workspace 窗口 然后选择编译)
2. 选择 [Build]→[Build] build 目标. (等同于 右击 工程名在 Workspace 窗口 然后选择 Build)
3. 选择 [Build]→[Rebuild All] rebuild 工程及工程相关配置
4. 选择 [Build]→[Stop Build] 停止编译连接

Build 信息显示在输出窗口的 Build 页。编译若没有错误,则编译通过, you can also use μ'nSP® IDE to 完成其他操作, 但是一些菜单命令和工具条按钮可能无效。在执行停 Build 之前, μ'nSP® IDE 首先关闭当前命令。如果当前命令无法停止,系统将一直等待当前命令执行完。

运行程序(RUN PROGRAM)

build 完成工程后,用户就可以开始应用程序,或运行应用程序,或下载到集成调试器。选择 [Build]→[Start Debug]→[Download] 下载可执行程序, 然后, 用 Debug 菜单其它命令对其进行运行和调试。

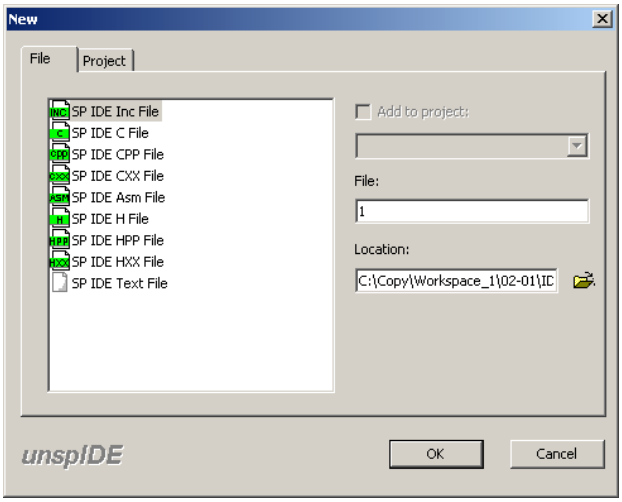
下载程序(LOAD PROGRAM)

选择 [File]→[Load Program]下载可执行文件(*.bin, *.TSK, *.S37) 然后调试/运行.

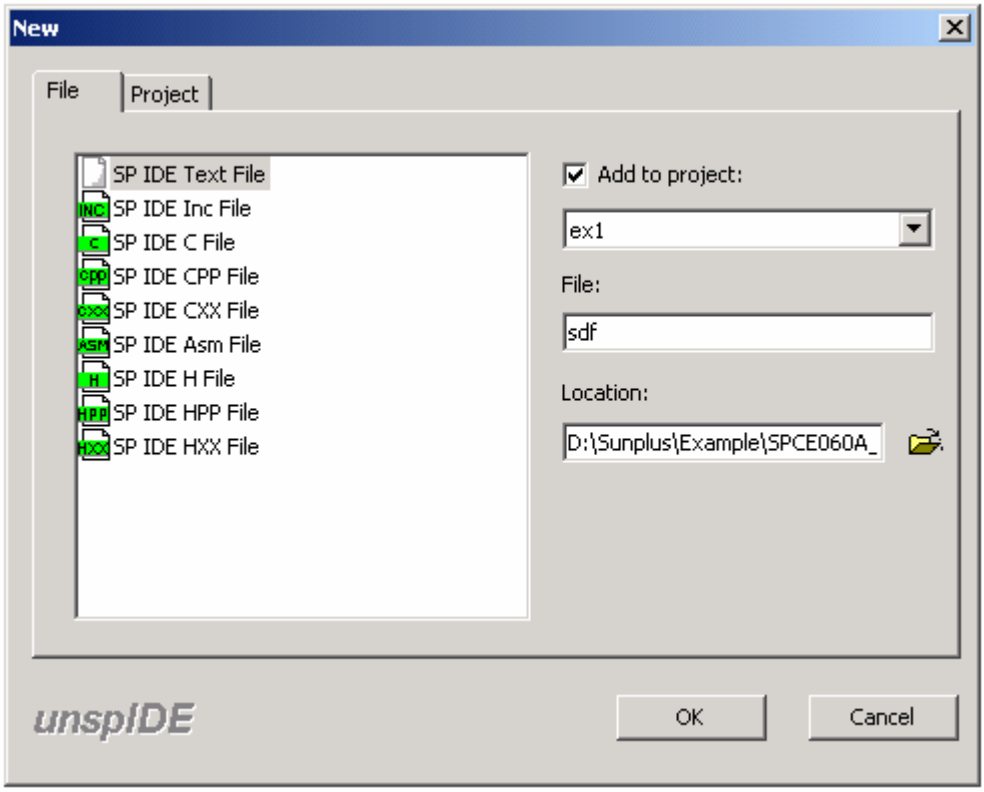
23.6源代码

23.6.1 创建文档

工具支持 Inc(包含文件), Asm(汇编文件), C, CPP, CXX, H, HPP, HXX, 文本文件。选择[File]→[New]打开 "New" 对话框:



单击文件条，在列表中选择文件类型。 添加文件到工程, 选择 Add to project 光标至File 框并键入文件名 File 框。如下图:



在 Location 框中设定文件路径并单击 [OK]。

23.6.2 打开文档

选择 [File]→[Open] 显示 Open 对话框。打开文件类型可以选 binary/text。打开文件另一种方法：选择 [File]→[Recent Files] 选择文件直接打开。最近打开文件列表最多 8 个近来访问的文件。

23.6.3 保存文档

如果一个文件名称标题条上有"*" 意味正在编辑当前文件. 若用关闭文件,"*"消失, 编辑器会提醒保存文件。 一旦保存了文件, "*" 将消失。

1. 选择[File]→[Save] 保存文件。
2. 选择 [File]→[Save As]将文件保存为 。
3. 选择[File]→[Save All] 保存所用打开的文件。

23.6.4 书签

用户可以在文本编辑器中自己文件放置书签。书签在文件关闭时自动取消。

1. 设置书签

(1) 移动光标到欲放置书签处 然后选择[Edit]→[Bookmark] 添加此书签。用户可用 Ctrl+n(n=0-9) 设置书签的个数。

(2) 选择 [Edit]→[Previous Bookmark] 移动光标至上一个书签处。

(3) 选择 [Edit]→[Next Bookmark] 移动光标至下一个书签处。

2. 删除书签

(1) 移动光标到欲清除书签处 然后选择[Edit]→[Bookmark] 清除设置的书签。

(2) 选择 [Edit]→[Clear All Bookmark] 清除所有设置的书签。

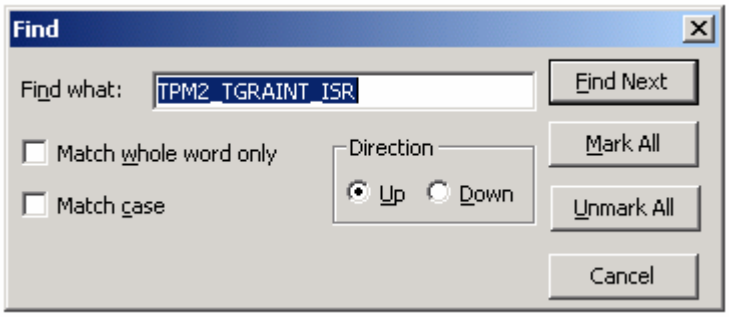
(3) 关闭文件将清除所有书签。

23.6.5 字符串查找

文本编辑器提供三种查找字符串的方法:

1. 在打开文档中查找文本

(1) 选择 [Edit]→[Find] 打开 "Find" 对话框。如下图:



(2) 在 "Find What" 框内输入要查找的字符串。

(3) 选择 "Match Case"/"Match Whole Word Only" 设置查找是部分匹配/完整匹配。

(4) 选择 "Up"/"Down" 决定是从光标到文件头还是光标处到文件尾查找。

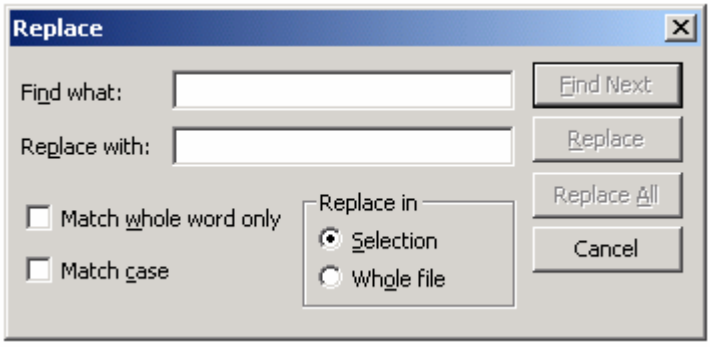
(5) 单击 [Mark All] 用书签表明查找匹配处, 单击 [Unmark All] 取消查找匹配书签。

(6) 单击 [Find Next] 查找匹配字符串。

(7) 选择 [Edit]→[Find Next] 和 [Find Previous]查找下一个或下一个匹配字符串。

2. 字符串替换

(1)选择 [Edit]→[Replace]打开 "Replace" 对话框。如下图:



(2) 输入要查找的字符串 A(string A)到 "Find What" 框。 输入用来替换的字符串 B (string B)"Replace With" 框, 若"Replace With" 框为空,编辑器就删除查找到字符串。

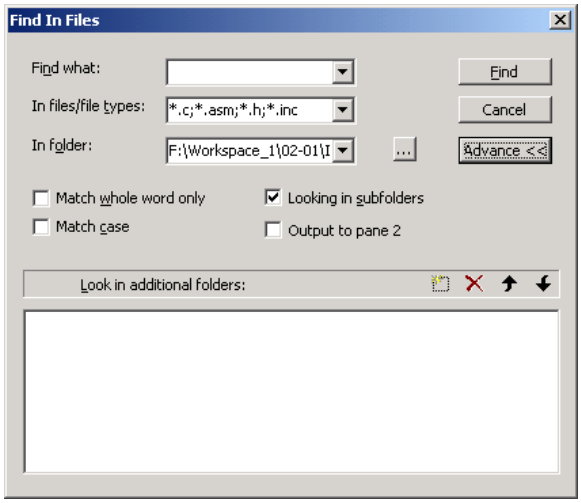
(3) 选择 "Match Case"/"Match Whole Word Only" 设置查找是部分/单词匹配。

(4) Selection/Whole file: 仅在当前匹配处替换/ 整个文件中匹配替换。Selection 选项在没有匹配时无效。

(5) 单击 [Find Next]忽视当前匹配的字串. 单击 [Replace]/[Replace All] 以替换当前匹配的/整个文件中匹配的。

3. 在磁盘文件中搜索字符串

(1) 选择 [Edit]→[Find In Files] 以显示 "Find in Files" 对话框。



(2) 输入字符串到 "Find What" 框。

(3) 在 In files/file types 下拉列表中选择文件类型。

(4) 在 "In folder" 框输入路径或单击 "..." 按钮以浏览路径。

(5) 选择 "Match whole word only"或"Match case" 以指定字串是整个字匹配或关键部分匹配。选择 "Looking in subfolders" 在 subfolder (默认选项)查找. 选择 "Output to pane 2"在输出窗口 Files2 查找页上输出查找结果.否则在输出窗口 Files1 查找页上输出查找结果。

(6) 选择 [Advance>>] 已指定查找的路径。

(7) 单击 [Find]。

23.6.6 二进制编辑

1. 打开进制文件

(1) 选择[File]→[Open] 以弹出 "Open"对话框. 等同选择[File]→[Recent Files]。

(2) 在 "Open as "框中选择 "Binary"。

(3) 在文件列表中选中文件。

(4) 点击[OK]。

注意：如果另一个编辑器正在编辑资源（文件），用户在使用二进制编辑器前应关闭它。

2. 编辑二进制数据

(1) 在二进制编辑器选择 HEX 段或 ASCII 段。键入 [Up]、[Down]、[Left]、[Right]、[Page Up]、[Page Down]、[Home]、[End] 在二进制编辑器以执行。

(2) 修改代码段内容。

(3) 保存修改。

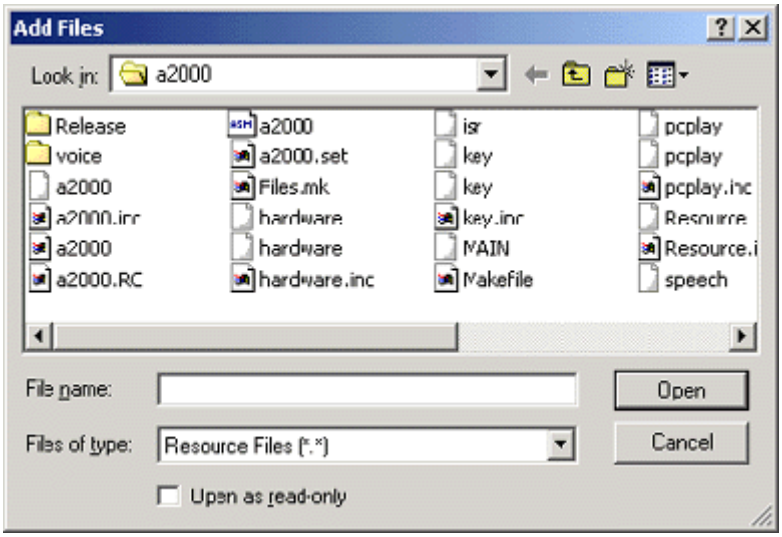
23.7 资源

23.7.1 资源 ID

资源 ID 包含一串字串。资源加入后，默认的名字为 RES_* (where * is the filename) 同时资源 ID 也被分配。

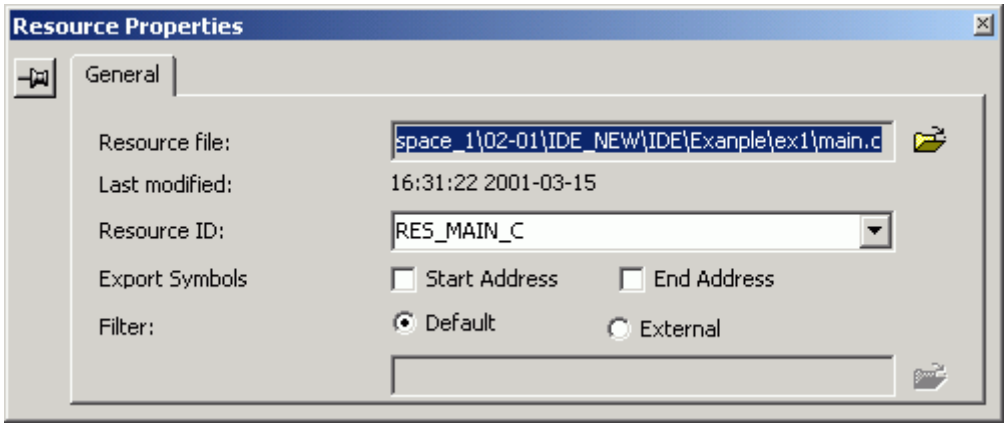
23.7.2 添加资源

在 Workspace 窗口的资源页，选择一组，右击，显示热键菜单。选择[Add Files to Folder] 以添加资源。The "Add Files" 对话框如下图所示:



23.7.3 改变工程内的资源

在工程区窗口的资源页，选中资源（文件）,右击,热键菜单如下图所示:



[Open]: 在二进制编辑窗口选定资源

[External Editor]: 用外部编辑器打开资源文件

[Remove]: 从当前工程中移出所选资源

[Hide]: 隐藏 Workspace 窗口

[Properties]: 改变所选资源文件的位置，资源 ID，输出符号,选择相应的外部滤波器。连接前，用户可以选择外部滤波器转换资源文件

外部滤波器格式如下：Filter.exe <input file> <*output.sp> *输出的扩展名 ".sp"

23.8 调试器

集成调试器可以帮助用户在语法错误清除完后，查找(锁定)逻辑错误。μ'nSP® IDE 能够监测程序运行到哪儿或停止程序到指定点以进一步分析，等所有变量,存储器 和寄存器的内容都能在调试模式下显示。

23.8.1 运行控制

首先，选择[Build]→[Start Debug]→[Download] 以下载程序，开始在 debug 模式下调试程序。一旦调试程序，用户可以使用 Debug 菜单的命令对程序进行如下的控制。下面的控制表列出了所有命令和详细描述：

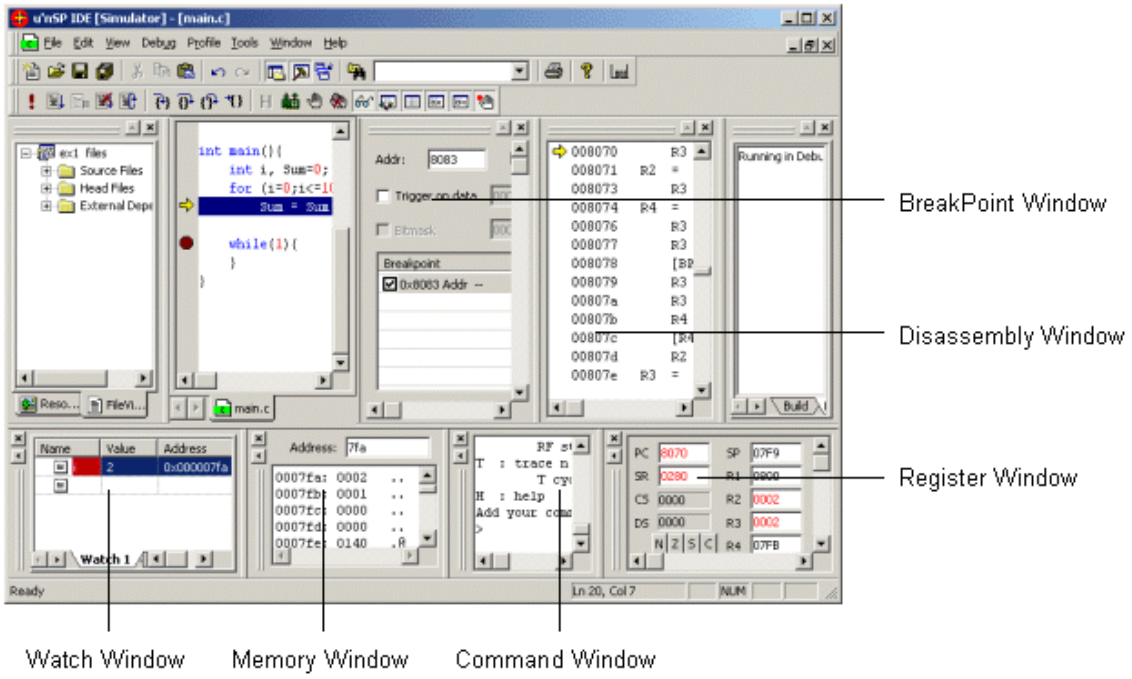
菜单	功能
Go(F5)	从当前位置运行到第一个断点或程序结束
Restart(Ctrl+Shift+F5)	重新开始运行程序
Stop Debug(Shift+F5)	停止调试程序，返回通常编辑状态，系统自动保存断点以便下次运行时恢复
Break	暂停
Step Into(F11)	单步进入调用函数
Step Over(F10)	单步越过下一个调用函数

菜单	功能
Step Out(Shift+F11)	从调用函数中返回，继续下一个指令,用 Step Into 和 Step Out 指令用户可以跟踪部分源代码
Step to Cursor(Ctrl+F10)	从当前命令行到光标处,如同在光标处设一个断点

用户可以单步越过指定代码继续调试程序. 用户也可以改变变量再运行。此外, 系统允许运行整个程序或单步运行一部分。

23.8.2 调试窗口

在调试模式下，调试菜单会显示在主菜单条上系统会用一系列窗口显示调试信息。用户可以使用 View 菜单命令访问这些窗口。



Memory 窗口： 显示存储器内容

Register 窗口： 显示通用寄存器和 CPU 状态寄存器当前内容.

Command 窗口： 显示命令的窗口.

BreakPoint 窗口： 显示断点内容.

Watch 窗口： 显示变量或表达式的内容. 允许输入和编辑表达式.

Disassembly 窗口： 显示存储器反汇编内容.

History Buffer 窗口： 显示执行的指令，状态和存储器内容

23.9代码分析器

23.9.1 代码分析器功能

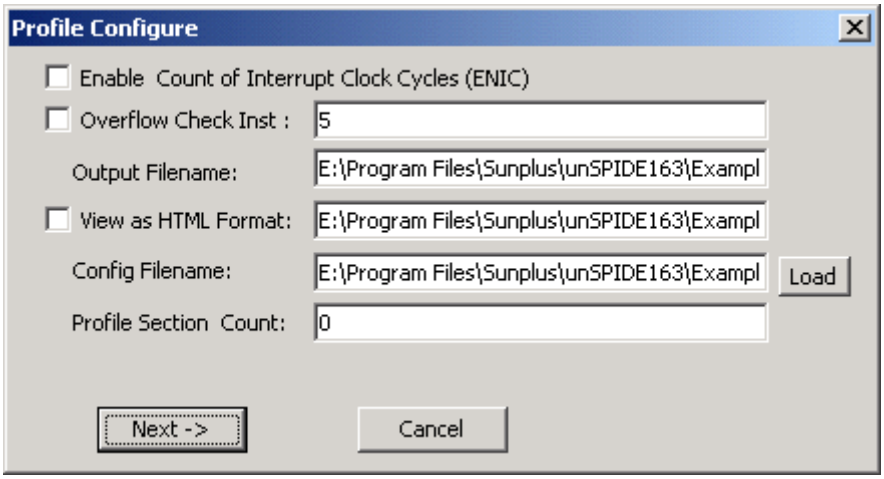
代码分析器是 μ'nSP® IDE 强有力的使程序有效率的分析工具。代码分析器功能如下所示:

1. 为最好代码使用提供信息。
2. 监测运行程序, 测量其算法是否有效率。
3. 由测试软件监测是否某些代码段写溢出。

此外，μ'nSP® IDE 提供 profile 允许对部分程序分析的功能。分析包括指令周期, IRQ, 流程 (label flow) 分析一些重要信息以协助提高程序品质。运行 profile 之前, 用户需停止程序的运行和调试。选择 [Debug]→[Stop debug]以停止调试。

23.9.2 代码分析器操作

1. 在调试模式选择[Debug]→[Profile]或直接选择[Profile]，将显示如下对话框:



使能中断时钟周期 (ENIC): ENIC 用于在 IRQ 中继续分析。如果 profile 部分 (a program section being analyzed) 位于 IRQ 中, 需使能选项

Overflow Check Inst: 如果不选 overflow 标志.,将设置报警的指令数

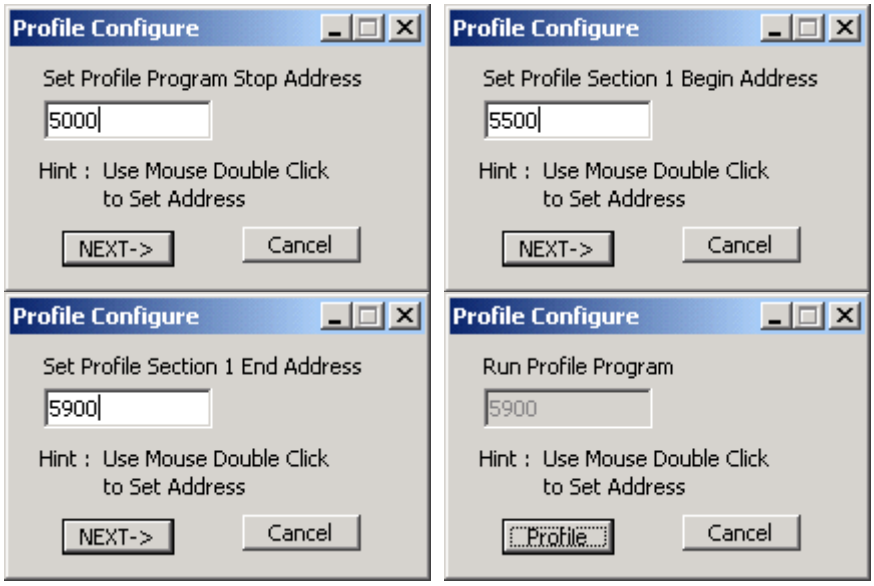
Output Filename: 最终的文件名

Config Filename: 设置可以保存于文件,以后可重新加载

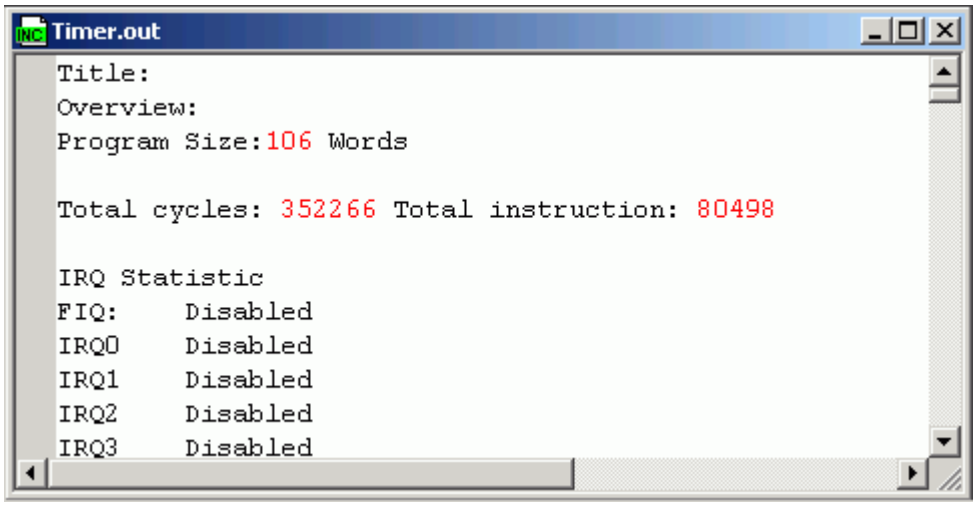
Profile section count: 被分析部分数。N 不得大于 32。 一个 Profile 部分就是程序被分析的部分

2. 在分析配置框中单击 [Next] 以附加对话框

在这个对话框, 拥护需指定分析部分的结束地址。用户可以键入指定地址到对话框也可双击指令条。同样, 用户需要指定分析部分的结束地址和开始地址。



3. 选择[Profile] 以进行分析。产生一个 profile 文件。用户从分析运行一直到停止,都可以在屏幕查看 profile 文件



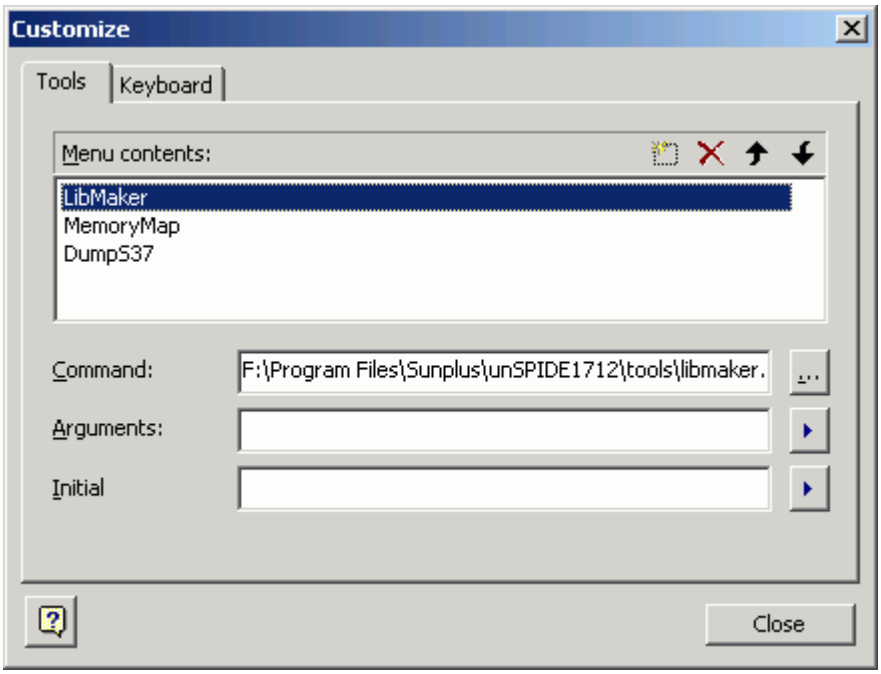
23.10 工具栏

23.10.1 定制用户开发环境

应用工作越多就越倾向定制一个 μ'nSP® IDE 舒适有效的环境。The μ'nSP® IDE 实际上提供达到的路径。例如，用户可以设置热键在 μ'nSP® IDE 中调用外部程序。一旦设置完成系统会保存设置。

选择[Tools]→[Customize]便出现如下对话框：

[Tools]：



这提供增加外部应用。那么工具栏菜单 user-defined 工具使能，用户可以在当前的 IDE 下运行。仅单击工具栏菜单应用, μ'nSP® IDE 自动调用相关应用。

用户可以在工具菜单上添加命令。这些命令被连接到在用户操作系统上运行的程序。用户可以在工具菜单对命令添加参数。

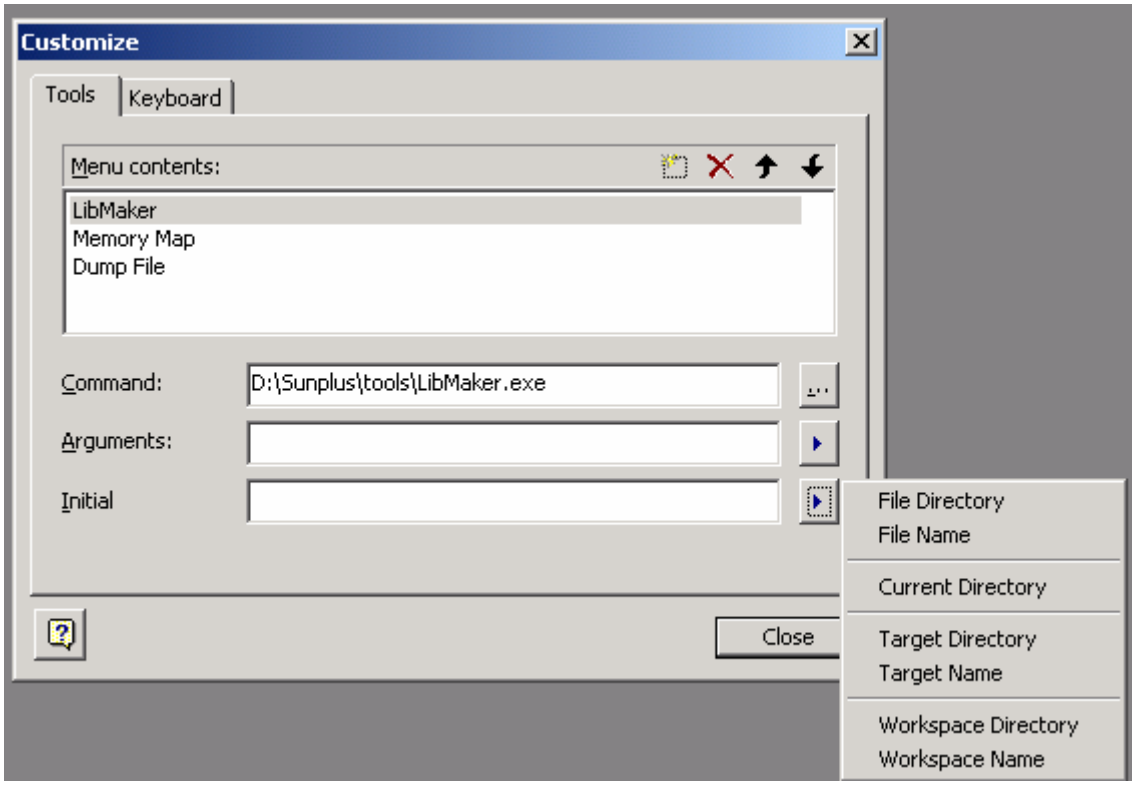
1. 在工具菜单上添加命令

   (1) 在菜单内容框,添加一个工具，单击 以添加一个新的内容。用户也可以双击列表中空白线(indicated by an empty rectangle)，键入欲添加的名字并显示出来，然后单击 [ENTER]。高亮的工具名字就是新添加的。

   (2) 在 Command 框内输入可执行程序的路径和程序名称，也可通过浏览按钮来设置。例如： C：\WINDOWS\NOTEPAD.EXE。

   (3) 在 Arguments 框内输入传递给外部工具所使用的参数。

   (4) 在 Initial 框内输入可执行程序的默认路径。



通过对 Menu Text 进行编辑，用户可改变工具的标识。

Arguments 和 Initial Directory 的各个参数如下：

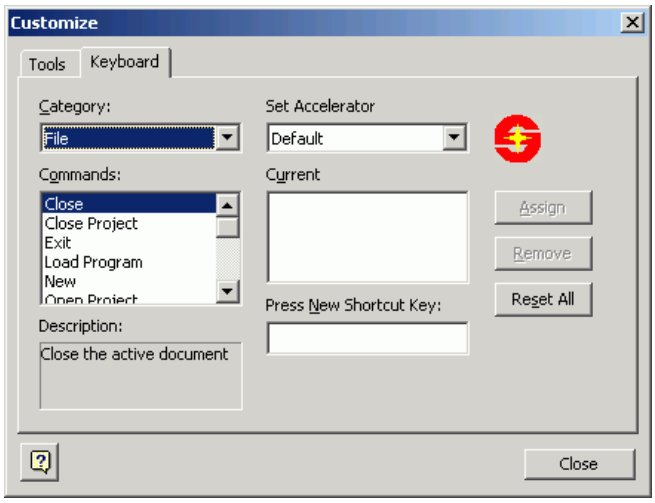
菜单内容	Argument 宏	描述
File Path	$(FilePath)	当前资料全名(defined as drive+path+filename)；如果激活一个 nonsource 窗口，那么为空
File Directory	$(FileDir)	当前资料目录(defined as drive+path)；如果激活一个 nonsource 窗口，那么为空
File Name	$(FileName)	当前资料名称(defined as filename)；如果激活一个 nonsource 窗口，那么为空
File Extension	$(FileExt	当前资料扩展名称
Current Line	$(CurLine)	当前激活窗口的 cursor line 位置
Current Column	$(CurCol)	当前激活窗口的 cursor column 位置
Current Text	$(CurText)	当前文本（位于当前光标下的字或只有一个字时选中一行）
Current	$(CurDir)	当前工作的目录（defined as drive+path）

菜单内容	Argument 宏	描述
Directory		
Target Path	$(TargetPath)	当前目标的全名(defined as drive + path + filename)
Target Directory	$(TargetDir)	当前目标的目录(defined as drive+path)
Target Name	$(TargetName)	当前目标的文件名(defined as filename)
Target Extension	$(TargetExt)	当前目标的扩展名
Workspace Directory	$(WkspDir)	当前工作区目录 (defined as drive + path)；如果当前没有工作窗口（workspace），那么为空
Workspace Name	$(WkspName)	当前工作区名(defined as filename)；如果当前没有工作窗口（workspace），那么为空
Project Directory	$(ProjectDir)	当前工程目录(defined as drive + path)
Current Body	$(CurrentBody)	当前 body 类型

3. 改变 Tool 菜单上的工具

在 **Menu Contents** 框内，选择要改变的工具，然后，用户可改变工具标识和相应的设定。利用在 **Menu Contents** 框上部的各个按钮，用户可改变工具在 Tool 菜单上的位置，也可以删除工具。

[keyboard]：



在这个窗口用户可以通过键盘对指定命令设置热键。用户可以删除改变键的配置，用

户可以为每个编辑器分配热键。用户还可以恢复所有热键到默认设置。

1. 分配和删除热键

   在分类列表中,选择欲设置热键的包含命令的菜单。在命令列表中,选择欲设置为热键的命令。在 Press New Shortcut Key 框中键入欲设的键或组合键，然后单击 [Assign]。如果按下的键或组合键无效，则没有键显示。 用户不得分配 ESC,键 和 如 CTRL+ALT+DEL 的组合键 ,这些已被操作系统所用。在当前键列表中， 选择欲删除的热键,然后单击 [Remove] 以删除热键。

2. 恢复热键到默认设置

   恢复热键到默认设置,只用单击 [Reset]。

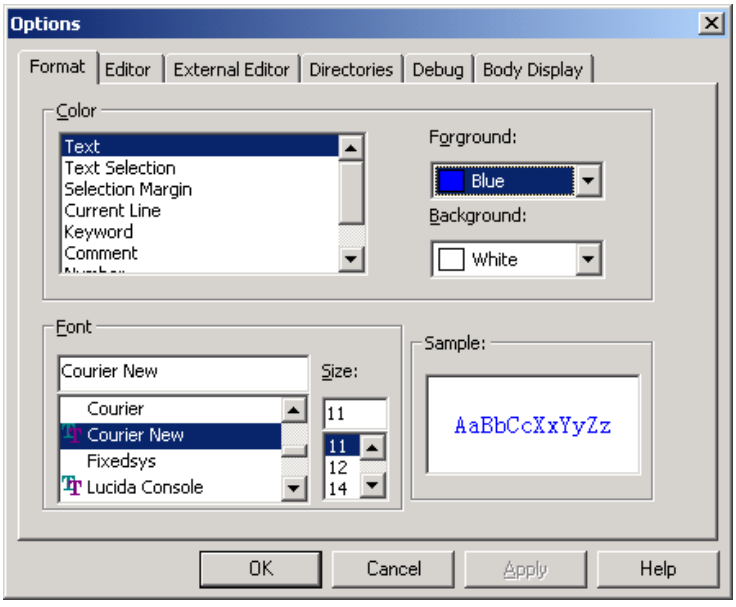
23.10.2 自定义编辑器

选项对话框包含设置 μ'nSP® IDE 编辑器的形式和功能， 包含文件和库文件路径用于编译和连接模式，前景颜色，不同窗口的正文颜色等。

在选项对话框，Format label 只能用在文本编辑器。许多设置能直接影响编译器和调试器的功能。当前显示模式和调试器状态确定其它一些设置。用户显示界面有四种显示布局:

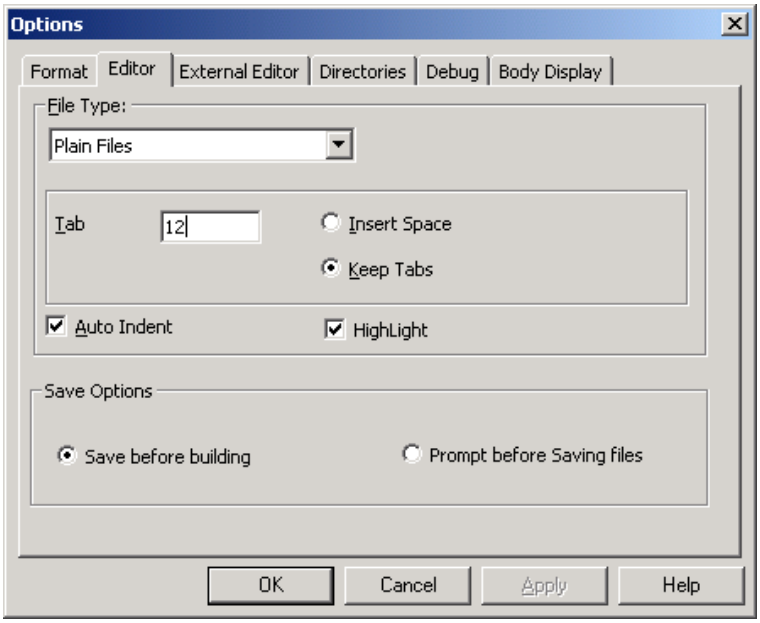
1. 在编辑模式下标准布局。
2. 在编辑模式下满屏布局。
3. 在调试模式下标准布局。
4. 在调试模式下满屏布局。

[Format]



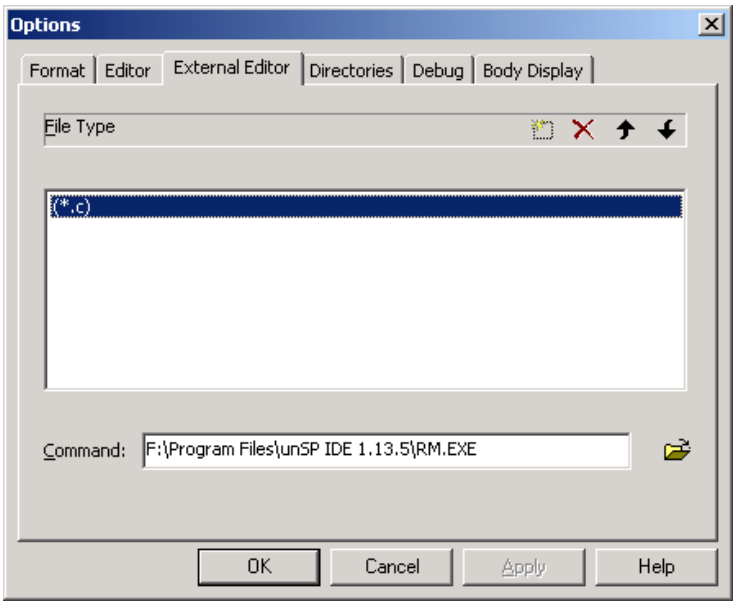
不同的元素(如 关键字, 注释 和 数量)可以设置为不同的字体,大小,前景, 背景颜色

[Editor]



用户可以依据对应文件的类型设置 tab 键的大小，tab 模式 和 缩进模式。选择 Save before building 以指定 IDE 在 build 工程之前保存文件。选择 Prompt before Saving files 以指定保存文件之前显示提示以便用户确认要保存的文件。

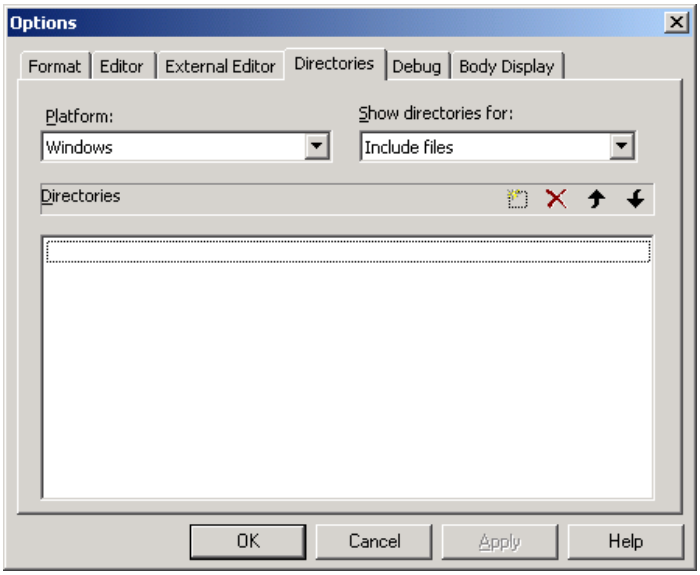
[External Editor]



在此窗口安装外部编辑器，用户可以用 Ctrl + E (hot key)使用外部编辑器编辑选择和激活的文件。单击 以创建一个条目。如，在列表框键入正确的文件类型如：c 文件类型(*.c)等。

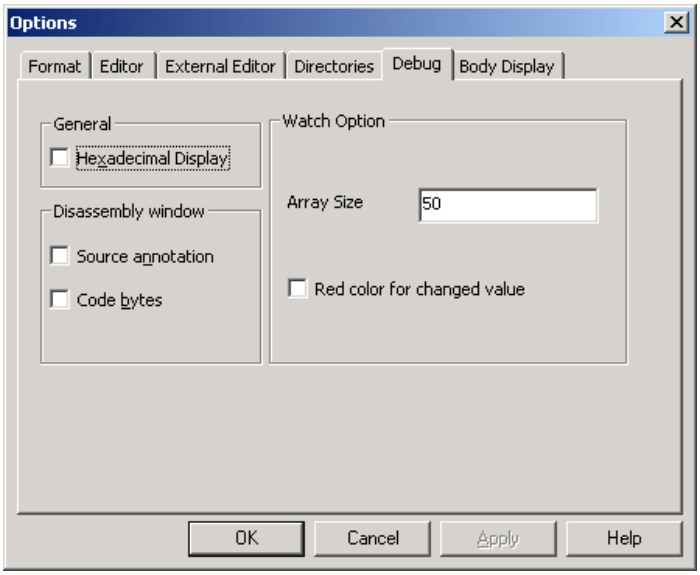
在命令区，用户可以自己选择的文件选择编译器。

[Directories]



选择平台路径。在路径列表中选择包含文件和库文件的路径。

[Debug]



Hexadecimal Display： 以十六进制格式显示数值，以十六进制列出用户在调试和对话窗口的输入。

Source annotation： 在 Disassembly 窗口内显示源代码。

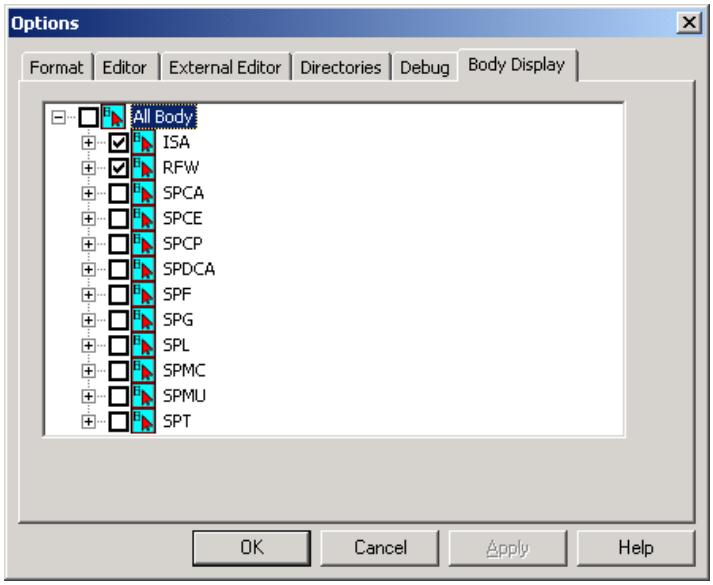
Code bytes： 在 Disassembly 窗口内显示指令的操作码。

Array Size： 指定在 Watch 窗口内显示的数组的长度。这个值的大小对调试速度有影响。

Red color for changed value： 指定 Watch 窗口中被改动的数据用红色显示，选中选项将影响调试的速度。

[Body Display]

指定工程支持的 body。

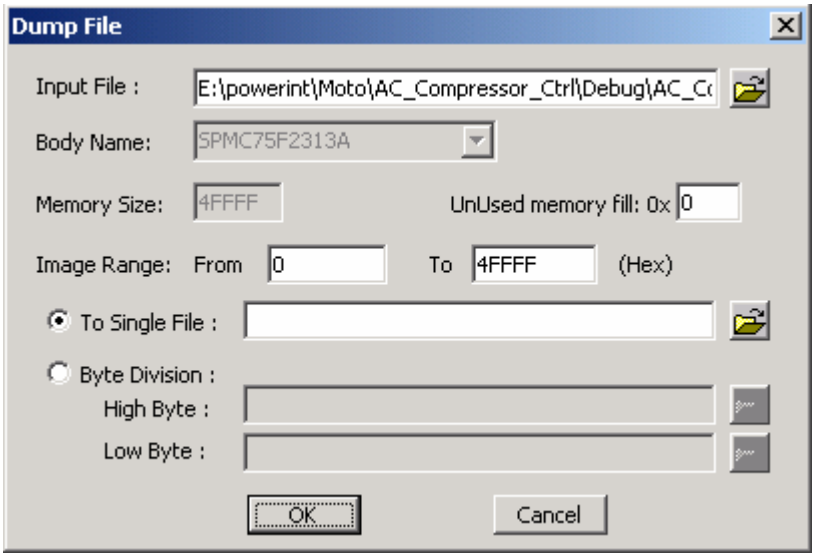


23.11 其他工具

23.11.1 Dump File

利用 Dump File 工具，可以把指定地址范围内的数据转存到文件内。激活 Dump File 工具的方法：选择 [Tools]→[Dump File]。可以看到一个如下窗口。输入的源文件位置，存储器的范围，输出文件类型选择，输入/输出文件名.单击[OK] 提取数据到指定文件。

注: 输入的源文件只支持 S37 & TSK 格式。



Input File: 打开源文件，源文件支持 S37 和 TSK 两种。

Body Name: 选择 body。

Memory Size: 显示存储器的大小。

UnUsed memory fill: 指定空白地址单元所填入的数据。

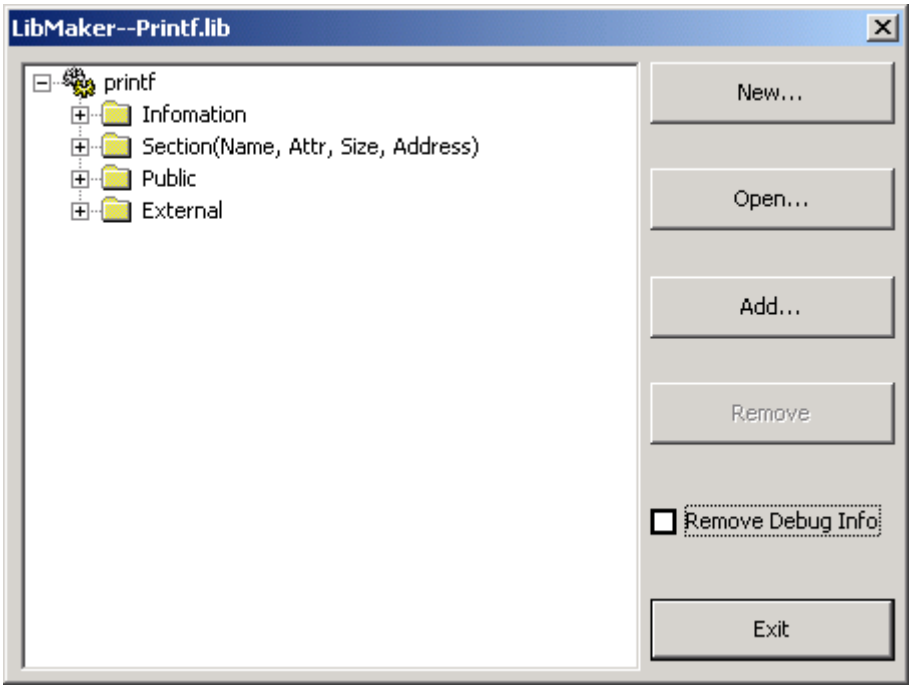
Image Range: 指定存储地址单元范围。

To Single File: 把指定存储地址单元范围内的数据转存到一个文件内。

Byte Division: 按高地址和低地址把指定存储地址单元范围内的数据转存到两个文件内。

23.11.2 LibMaker

选择[Tools]→[LibMaker], 将弹出如下图的一个创建库的对话框。用户可以在这里创建一个新库或打开一个存在的库,以添加或移出目标文件。



选择 [New]创建一个新库。

选择 [Open] 打开一个存在的库。

选择 [Add] 以添加目标文件到库。

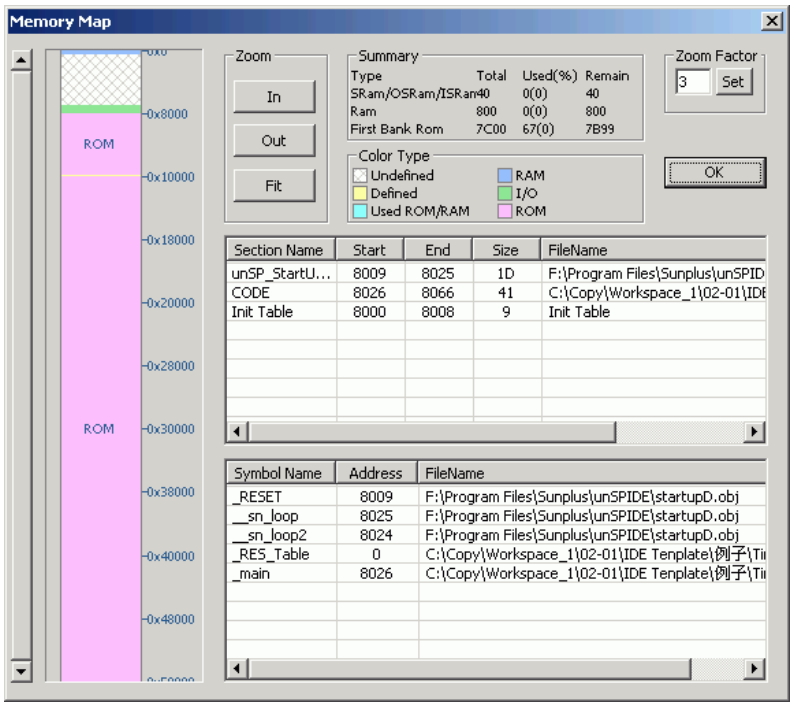
选中列表,然后按 [Remove] 将清除所选目标文件。

选中 Remove Debug Info 以清除目标的调试信息。

选择 [Exit] 退出此窗口，返回主窗口。

23.11.3 MemoryMap

选择[Tools]->[MemoryMap], 弹出如下图所示的 memory-map 对话框。用户可以浏览存储器的配置、段、 符号和统计表信息。



单击［In］放大 存储器 map。

单击［Out］缩小存储器 map。

单击［Fit］缩放到窗口大小。

[Zoom Factory] 设置放缩因数。

单击［OK］退出此窗口返回主窗口。

23.12 工具栏按钮

μ'NSP® IDE 所有工具栏按钮列表：

创建新文件	选择 BODY
打开文件	调整窗口重叠排列
保存激活的文件	调整窗口非重叠排列
保存全部打开的文件	关闭全部窗口
剪贴到剪贴板	激活下一个 UNDOCKED 窗口
复制到剪贴板	激活上一个 UNDOCKED 窗口
插入剪贴板内容	暂停
撤消最后一次行为	停止调试程序
重复前一次行为	程序复位
显示或隐藏 WORKSPACE 窗口	进入单步（STEP INTO ）
显示或隐藏输出（OUTPUT ）窗口	绕过单步（STEP OVER ）
当前打开窗口管理	跨出单步（STEP OUT）
在所有文件中查找指定文本	运行到当前光标处
在当前文件中查找指定文本	显示历史缓冲器

激活文档打印	设置仿真器（EMULATOR）
显示 IDE 版本信息	插入或清除断点
下载程序到 DEVICE	清除全部断点
编译文件	显示或隐藏监视（WATCH）窗口
编译和连接工程	显示或隐藏反汇编（DISASSEMBLY）窗口
停止编译和连接	显示或隐藏存储器（MEMORY）窗口
执行程序	显示或隐藏寄存器（REGISTER）窗口
下载程序	显示或隐藏命令（COMMAND）窗口
模拟器模式调试程序	显示或隐藏断点（BREAKPOINT）窗口
ICE 模式调试程序	

第24章 凌阳创新 DMC 工具介绍

24.1DMC 简介

欢迎使用凌阳创新科技公司为您提供的具有强大功能的 DMC (Digital Motor Control)工具，此 DMC 工具有实时监控两台变频电机的功能。友好的用户界面和下来式菜单可以更好的协助您简便,有效的进行开发应用。它主要有以下两点功能：

系统控制：可以单独设置两颗电机的控制参数,如转速、加速斜率、PID 调节中控制增益 Kp、Ki 等，可为用户提供多达 16 个参数。

监视系统：电机监控参数可以显示在控制窗口中，同样可以以图形的形式在图形窗口中显示。

24.2系统要求

当前版本 DMC 能够运行在 Windows98 或 Windows2000 操作系统上。对 PC 系统最小需求如下：

CPU 主频: 500M Hz

最小内存容量: 64 MB

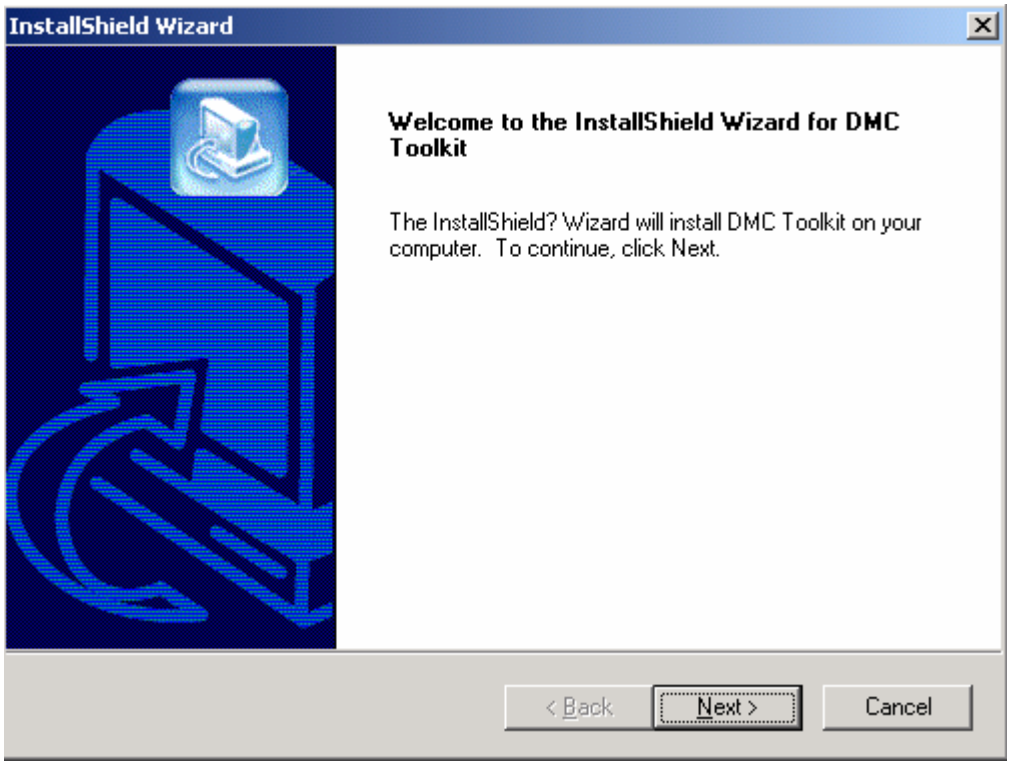
最小硬盘剩余空间: 20 MB

COM 串行接口

24.3安装指南

按照以下步骤安装：

1、执行安装文件

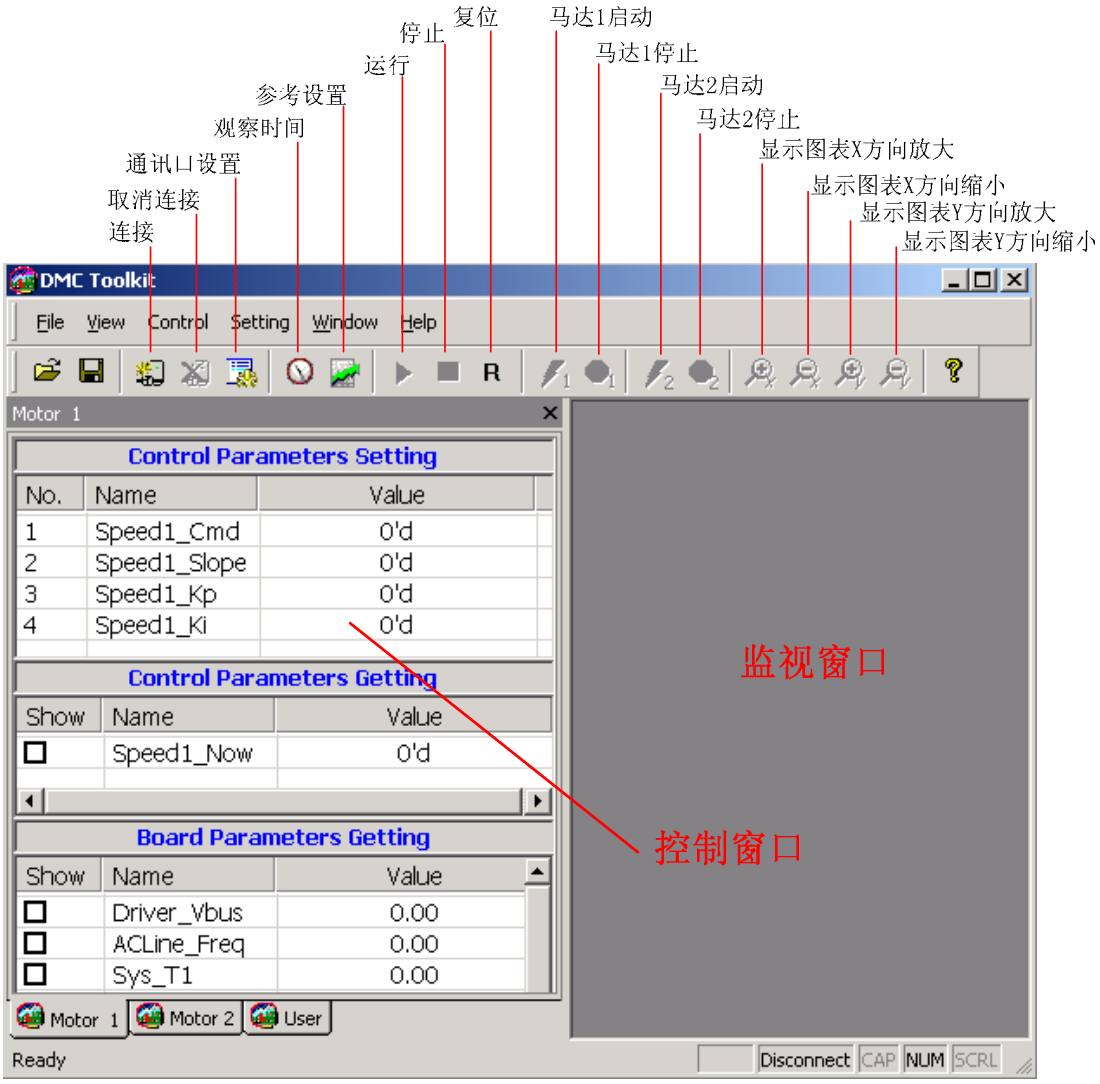


2、按照屏幕提示，完成安装。

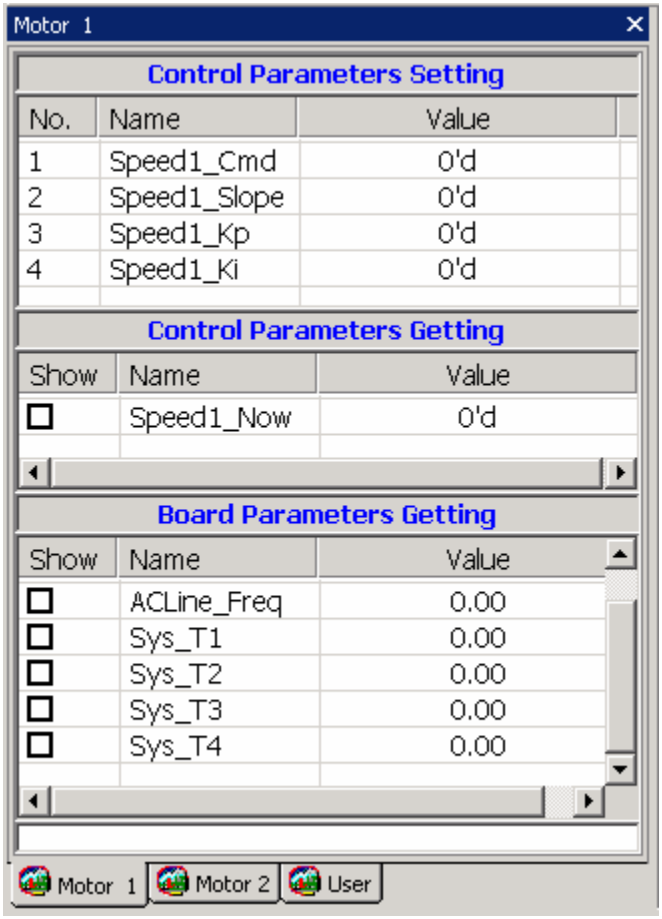
注：字体最好设置 小字体 (Normal Size (96 dpi)) 模式。

24.4DMC 工具组成

用户通过包括菜单、工具条、窗口对应用系统进行监控。DMC 工具包含两部分：控监视窗口，和图形窗口。


连接
取消连接
通讯口设置
观察时间
参考设置
运行
停止
复位
马达1启动
马达1停止
马达2启动
马达2停止
显示图表X方向放大
显示图表X方向缩小
显示图表Y方向放大
显示图表Y方向缩小
DMC Toolkit
File View Control Setting Window Help
Motor 1
Control Parameters Setting
No. Name Value
1 Speed1_Cmd 0'd
2 Speed1_Slope 0'd
3 Speed1_Kp 0'd
4 Speed1_Ki 0'd
Control Parameters Getting
Show Name Value
Speed1_Now 0'd
Board Parameters Getting
Show Name Value
Driver_Vbus 0.00
ACLine_Freq 0.00
Sys_T1 0.00
Motor 1
Motor 2
User
Ready
Disconnect
CAP
NUM
SCRL
监视窗口
控制窗口

24.5监控窗口



监控窗口用来设置/读取参数，由三部分组成：马达 1 监控窗口、马达 2 监控窗口、用户窗口。每种窗口由三部分内容组成：控制参数设定、控制参数读取、系统参数读取。各参数含义介绍如下：

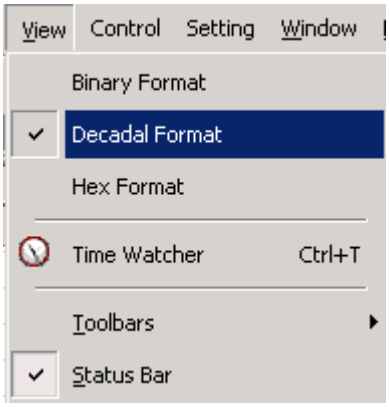
变量名称	属性	描述	范围
Speed1_Cmd	W	设定变频马达 1 转速(单位：rpm) (bit15 = 方向) 资料为 0 - 0x7FFF （共 15 bits）	**0-±0x7FFF** **（单位：rpm）**
Speed1_Slope	W	设定变频马达 1 加减速斜率(单位：rpm/sec) 资料为 0 - 0xFFFF （共 16 bits）	**0-0xFFFF** **（单位：rpm/sec）**
Speed1_Kp	W	设定变频马达 1 闭回路速度控制调适参数 Kp (只接受大于或等于零的常数输入）	**大于 0**
Speed1_Ki	W	设定变频马达 1 闭回路速度控制调适参数 Ki (只接受大于或等于零的常数输入）	**大于 0**
Speed2_Cmd	W	设定变频马达 2 转速(单位：rpm) (bit15 =方向) 资料为 0 - 0x7FFF （共 15 bits）	**0-±0x7FFF** **（单位：rpm）**
Speed2_Slope	W	设定变频马达 2 加减速斜率(rpm/sec) 资料为 0 - 0xFFFF （共 16 bits）	**0-0xFFFF** **（单位：rpm/sec）**

变量名称	属性	描述	范围
Speed2_Kp	W	设定变频马达 2 闭回路速度控制调适参数 Kp (只接受大于或等于零的常数输入)	大于 0
Speed2_Ki	W	设定变频马达 2 闭回路速度控制调适参数 Ki (只接受大于或等于零的常数输入)	大于 0
Speed1_Now	R	读取变频马达 1 现在转速(单位：rpm)含方向 (bit15 =方向) (bit 15 = 1 => 负转速, bit15 = 0 => 正转速) 资料为 0 - 0x7FFF （共 15 bits）	0-±0x7FFF （单位：rpm）
Speed2_Now	R	读取变频马达 2 现在转速(单位：rpm)含方向 (bit15 = direction) (bit 15 = 1 => 负转速, bit15 = 0 => 正转速) 资料为 0 - 0x7FFF （共 15 bits）	0-±0x7FFF （单位：rpm）
Driver_Vbus	R	读取驱动系统的直流链电压(单位：V)，此电压值只为正值 V >= 0 ：资料为 0 - 0xFFFF(in Q6) 例如: 若资料为 0x4D80 => 310 V	0 - 0xFFFF
ACLine_Freq	R	读取市电端电源频率(单位：Hz), 资料为 0 - 0xFFFF (in Q10) 例如: 若资料为 0xEE00 => 59.5 Hz	0 - 0xFFFF
Sys_T1	R	读取系统的温度信道 1(bit15 = 温度值正负号) 资料为 0 - 0x7FFF(in Q6) (单位：℃) 例如: 若资料为 0x3200 => +200.0 ℃ 若资料为 0xB200 => -200.0 ℃	0-±0x7FFF （单位：℃）
Sys_T2	R	读取系统的温度信道 2(bit15 = 温度值正负号) 例同上	0-±0x7FFF （单位：℃）
Sys_T3	R	读取系统的温度信道 3(bit15 = 温度值正负号) 例同上	0-±0x7FFF （单位：℃）
Sys_T4	R	读取系统的温度信道 4(bit15 = 温度值正负号) 例同上	0-±0x7FFF （单位：℃）
User_W0	W	用户区域(设定参)	
User_W1	W		
User_W2	W		
User_W3	W		
User_W4	W		
User_W5	W		
User_W6	W		
User_W7	W		
User_R0	R	用户区域(读取参数)	

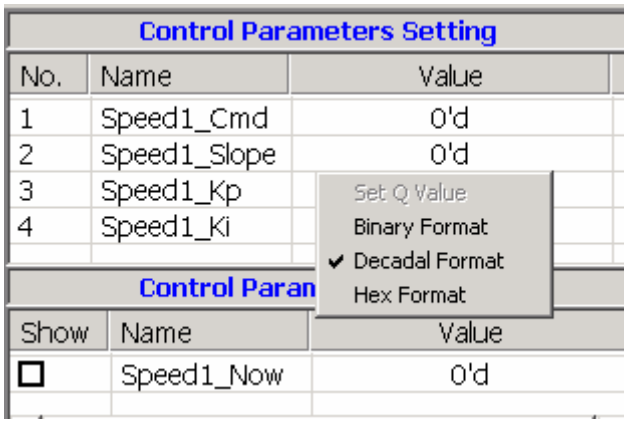
变量名称	属性	描述	范围
User_R1	R		
User_R2	R		
User_R3	R		
User_R4	R		
User_R5	R		
User_R6	R		
User_R7	R		

参数形式可以设置为十进制、十六进制或二进制显示，设置方法有两种：

（1）[View] → [Binary Format/Decadal Format/Hex Format]



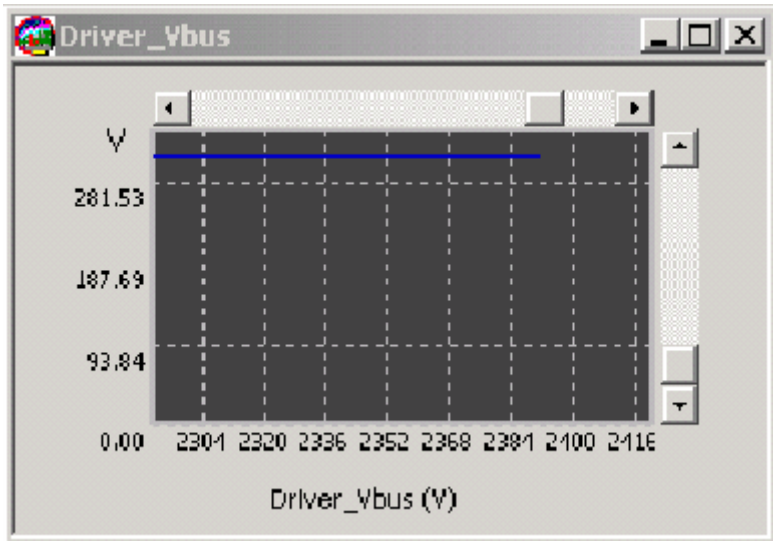
（2）在监制窗口右击鼠标。

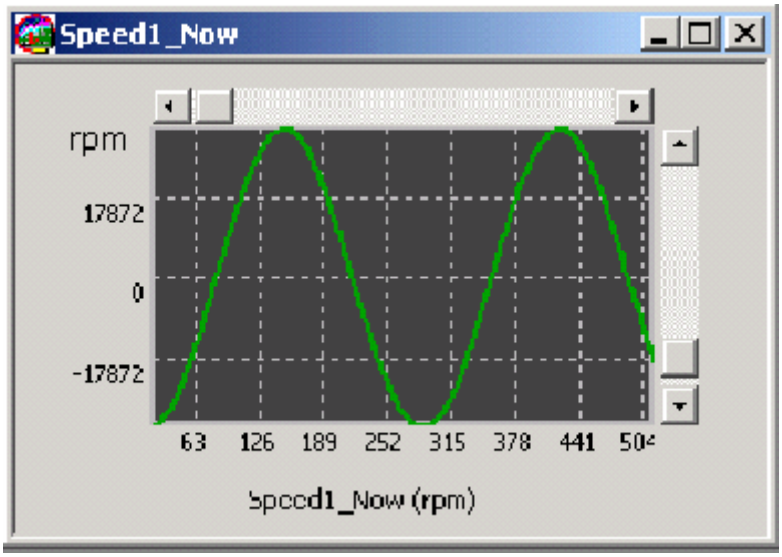


注：Q 格式数据只能应用于用户区；仅用户区的变量名称可以更改. 在变量名称上单击，修改名称，然后按回车或在界面的其他地方单击；修改参数值的方法同上。

24.6 图形窗口

图形窗口用来显示图形，用户勾选控制窗口内的变量来选择要显示的变量图形。在图形窗口显示的每个图形都是独立的，可以单独打开或关闭。



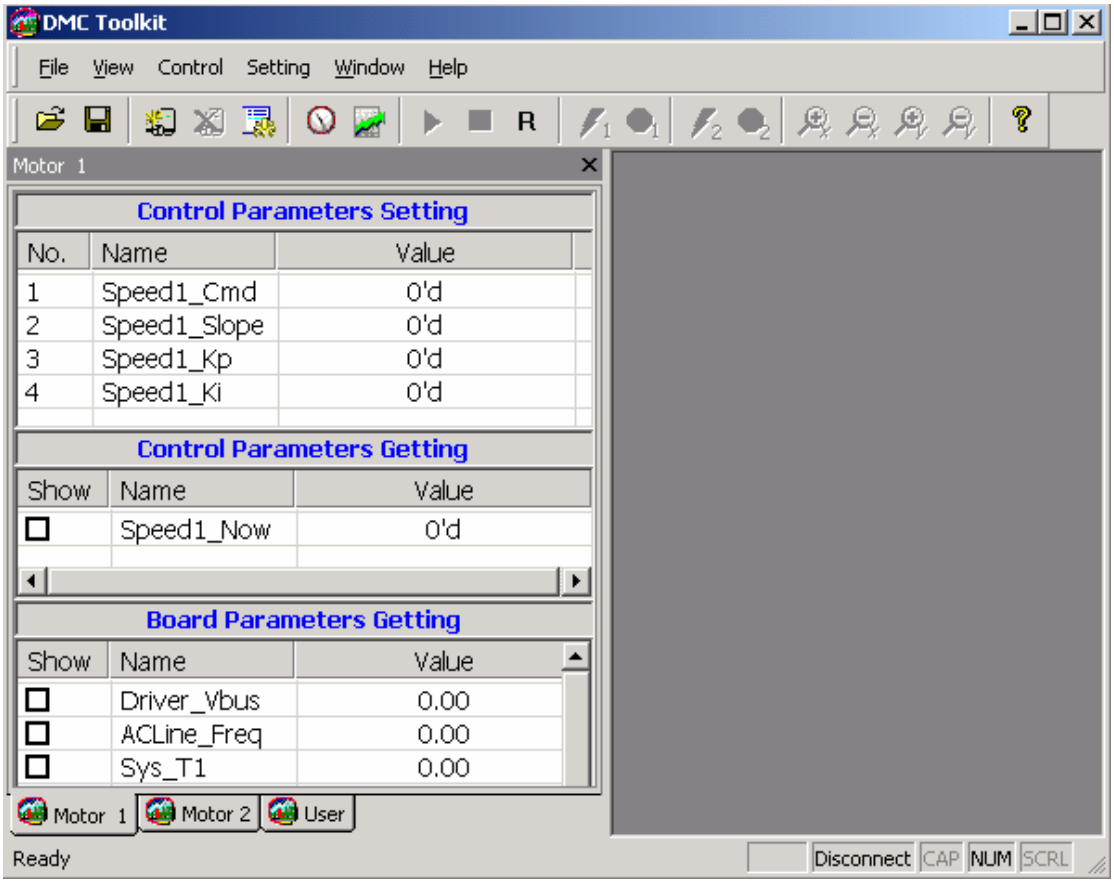


说明：用户可以通过设置 Window 菜单来安排窗口的排列方式；用户可以通过鼠标拖动来改变窗口的大小。

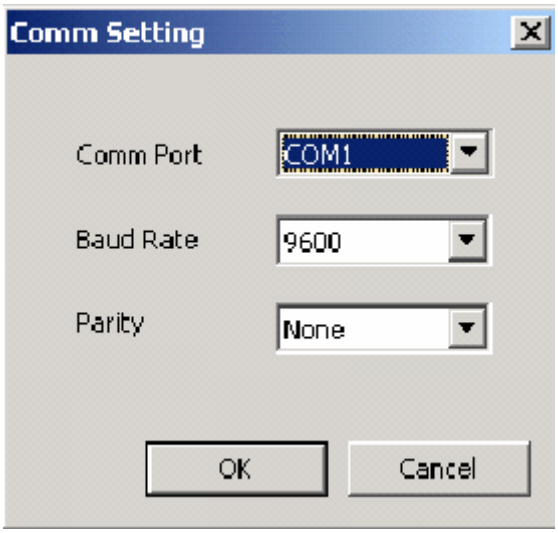
24.7 DMC 应用举例

24.7.1 启动 DMC 工具

1、选择[开始] → [程序] → [SunplusIT] → [DMC Toolkit] → [DMC Toolkit] 来启动 DMC 工具，启动后显示如下主界面：



2、选择 File] → [Comm Setting] 或单击 打开通讯设置选项。



3、选择 File] → [Connect]] 或单击 连接通讯界面。

可以选择 File] → [Disconnect]] 或单击 断开连接。

如果连接错误则弹出以下提示框。

24.7.2 系统监控

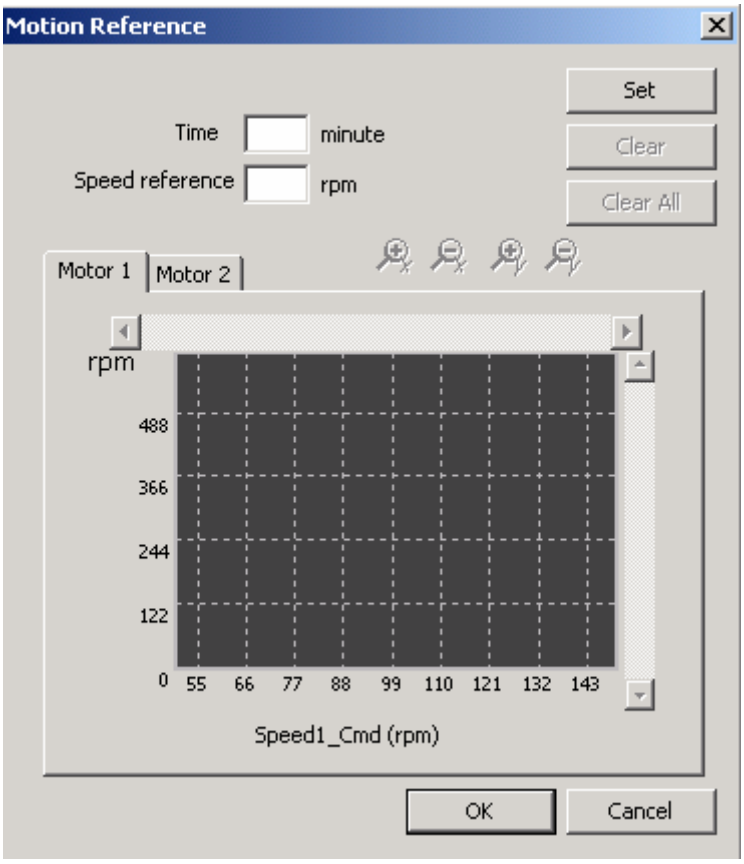
选择[Control] → [Start] 或单击 开始。

- 对系统进行控制

  有两种方式来设置控制参数。

  1、在控制窗口直接写入控制参数（参照 3.1 节）。

  2、选择 Setting] → [Motion Reference] 或单击 打开如下的参考设置界面。



Set：设置一个参考点；

Clear：清除一个参考点；

Clear All：清除所有参考点；

参考设置界面用来设置马达运行转速参数Speed_cmd

- 对系统进行监视

用户可以通过监制窗口来观察马达控制参数，也可以通过图形窗口观察控制参数波形。用户可以选择[Control] → [Stop/Reset]或单击 来控制。当按下Start，在图形窗口显示的图形会即时更新；当按下Stop，在图形窗口显示的图形停止更新；当按下Reset，在监视窗口显示的图形会复位。用户可以将数据导出为.txt格式的文件，导出方法为：在监视窗口显示的图形上右击，然后选择[Export…]。

24.8DMC 操作

- **启动/停止马达**

选择[Control] → [Motor 1 Start] / [Motor 1 Stop] / [Motor 2 Start] / [Motor 2 Stop]或单击 来启动/停止马达。

- **项目保存**

选择[File] → [Save Project] 或单击 来存储工程。也可以通过选择[File] → [Save As…]. 将工程以另外的名称命名或存到另外的目录。

- **退出**

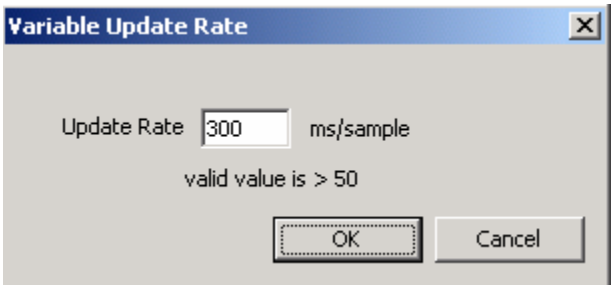
选择[File] → [Exit] 退出工具界面。

- **打开文件**

选择[File] → [Open] 或单击 来打开文件。

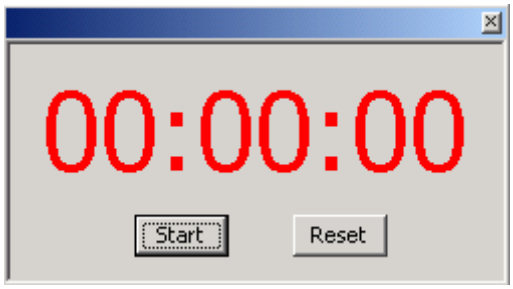
- **设置更新率**

在使用前可以通过选择 [Setting] → [Variable Update Rate…] 打开一个对话框来设置变量值的更新率，此设置对监控窗口的变量值栏和图形窗口的曲线都有效。



- **计时功能启动**

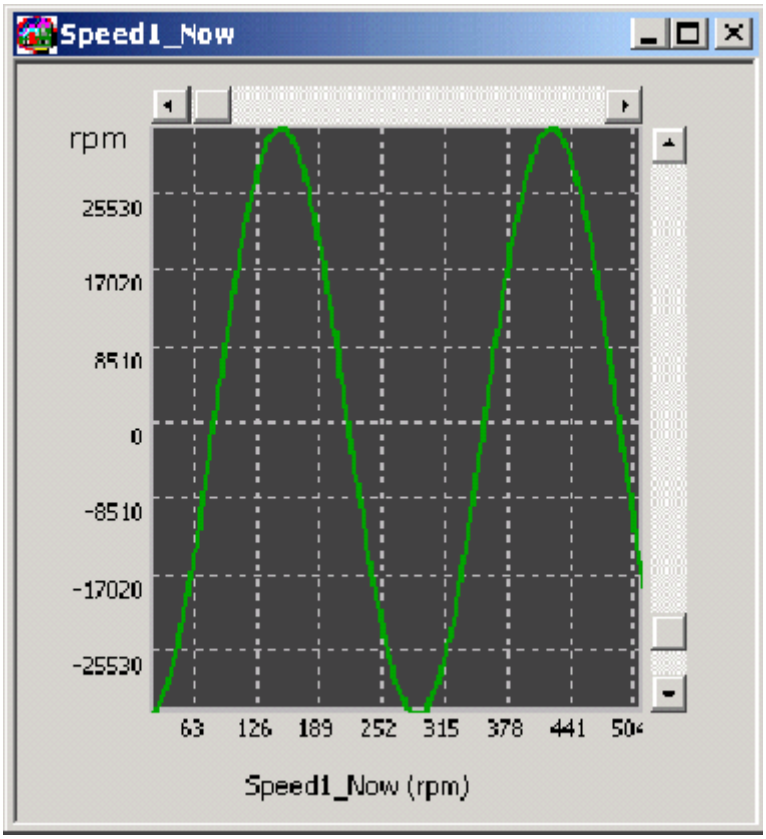
选择 [View] → [Timer Watcher]打开如下窗口。



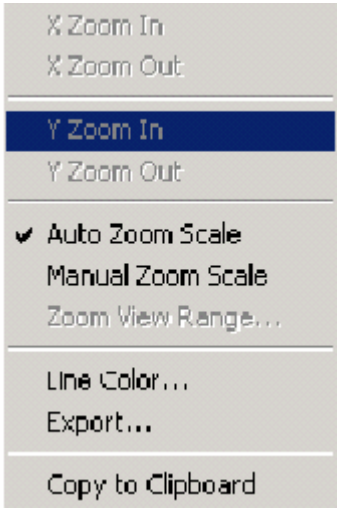
当按下 Start 键，计时功能开始，并且 Start 按钮会变成 Stop 功能。

说明：这里的开始、停止和马达运转没有关系。

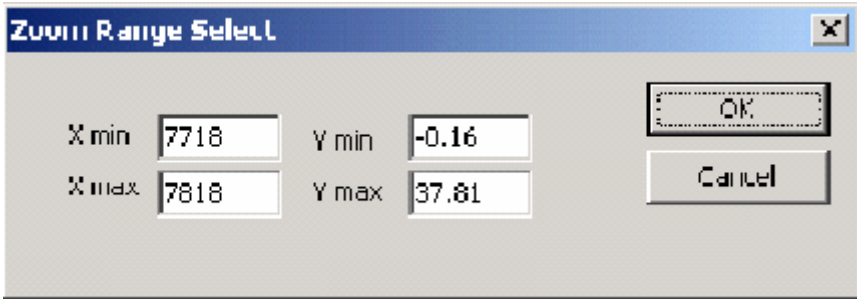
■ **编辑图形窗口**



在图形上右击显示如下界面：



在[Manual Zoom Scale]方式，可以通过选择[Zoom View Range…]打开[Zoom Range Select]窗口来设置观测范围。



[Line Color…]用来设置线的颜色，线的默认颜色为绿色。

[Export] 输出.TXT 格式的数据文件。

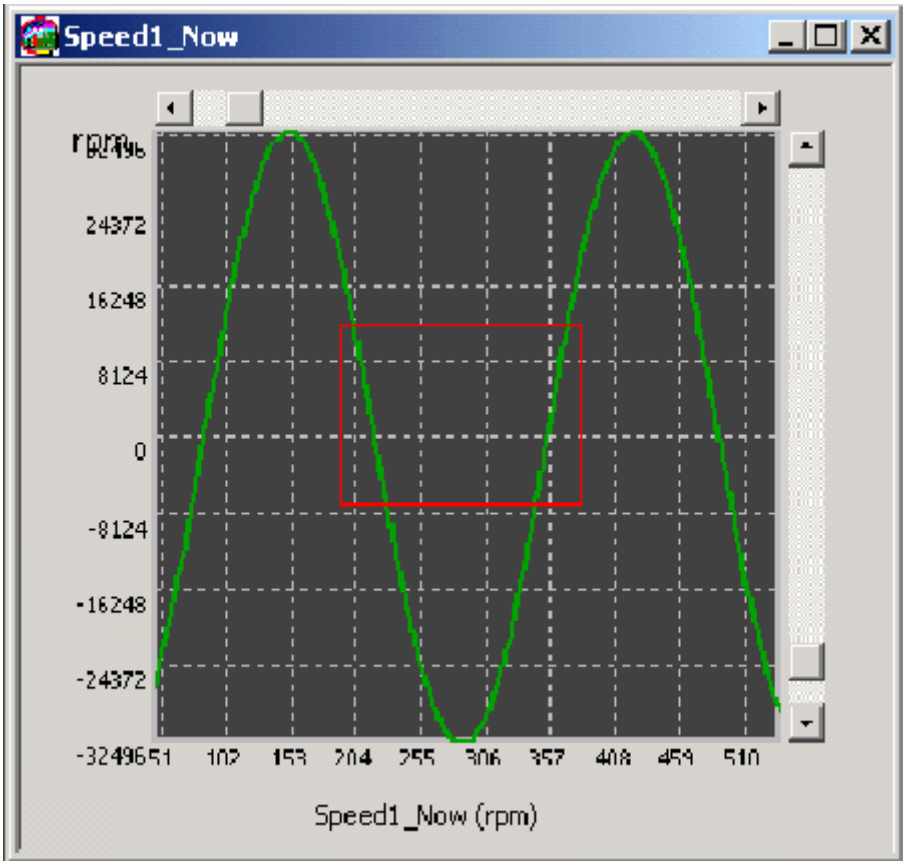
[Copy to Clipboard] 将图形拷贝到剪贴板。

说明：

1、用户可以通过[Setting] → [Chart Background…] 来改变图形窗口的背景颜色。

2、 用来放大、缩小图形。

3、用户可以通过拖拉鼠标的方式来放大图形，拖拉鼠标时红色区域的图形将显示在当前窗口当中。这种方式只能使用在[Manual Zoom Scale] 模式下。



■ **快捷方式**

Ctrl+O	打开工程
Ctrl+S	保存工程
Ctrl+T	监视时间
Ctrl+G	开始
Ctrl+P	停止
Ctrl+R	复位
Ctrl+M	打开参考设置界面

24.9技术支持

DMC 工具很容易使用,然而,某些时候您还是需要版本的介绍和说明,其实您也可以从 Help 得到帮助;帮助文件和 DMC 工具用户手册包含很多解答,此外您还可以利用 Web. 如果您还有疑问,可以通过 Email,得到我们及时的支持

您还可以直接访问

Web Site Support: http://www.SunplusIT.com/

Email Support: mike@SunplusIT.com

第25章 附录 A 保留字

除未列出的汇编语言命令外，下列字为μ'nSP™的保留字。

1. 有“__sn_”前缀的字被保留
2. BREAK, _FIQ, _RESET, _IRQ0, _IRQ1, _IRQ2, _IRQ3, _IRQ4, _IRQ5, _IRQ6, _IRQ7, __subf2, __blockcopy, __cmpf2, __common, __cvf2i1, __cvf2i2, __cvf2u1, __cvf2u2, __cvi1f2, __cvi2f2, __divf2, __divi1, __divi2, __divu1, __divu2, __lshiu1, __lshiu2, __modi1, __modi2, __modu1, __modu2, __mulf2, __muli2, __mulu1, __mulu2, __negf2, __rshi1, __rshi2, __rshu1, __rshu2, __addf2